SYSTEMS ENGINEERING
Research Center

Verification, Validation and Accreditation

Final Technical Report SERC-2011-TR-018

Principal Investigators:

Sue O'Brien, University of Alabama Huntsville

Russell Peak, Georgia Institute of Technology

Team Members - UAH
Philip Alldredge, Research Associate, University of Alabama Huntsville
Lance Warden, Research Scientist, University of Alabama Huntsville
Dr. Julie Fortune, Senior Research Engineer, University of Alabama Huntsville
Team Members—Research Professionals - GIT
Selcuk Cimtalay, PhD; Andy Scott; Miyako Wilson
Team Members—Undergraduate Research Assistants - GIT
Brian Aikens; Drew Martin

Contract Number: H98230-08-D-0171 **DO002, TO002, RT021**

Report No. SERC 2011-TR-018
May 3, 2011

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **03 MAY 2011**	2. REPORT TYPE	3. DATES COVERED **00-00-2011 to 00-00-2011**

4. TITLE AND SUBTITLE
Verification, Validation and Accreditation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Stevens Institute of Technology,Systems Engineering Research Center (SERC),1 Castle Point on Hudson,Hoboken,NJ,07030

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
SERC is sponsored by the Department of Defense.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**159**	

INTRODUCTION

This VV&A task (RT 21) had two parallel components which converged to provide recommendations on methods, languages and tools for most effective Modeling and Simulation (M&S) of complex systems as a means for Verification, Validation and Accreditation. One component, performed by University of Alabama Huntsville, is focused on an AADL approach, and exploits synergies with other related efforts being conducted by the SAVI consortium. The other component, performed by Georgia Institute of Technology, is focused on a SYSML based approach. Both components begin with building an understanding of the current M&S environment with a focus on gaps in M&S approaches, methods and tools being capable of supporting effective VV&A. A common assessment approach will inform both teams about the current environment and gaps.

This Final Technical Report is broken down into two components:

University of Alabama Huntsville – pages 3 – 33

Georgia Institute of Technology – pages 34 - 159

SYSTEMS ENGINEERING
Research Center

Verification, Validation and Accreditation using AADL

Principal Investigator: Sue O'Brien – University of Alabama Huntsville

Team Members

Philip Alldredge, Research Associate, University of Alabama Huntsville

Lance Warden, Research Scientist, University of Alabama Huntsville

Dr. Julie Fortune, Senior Research Engineer, University of Alabama Huntsville

Contract Number: H98230-08-D-0171 **DO002, TO002, RT021**

Report No. SERC 2011-TR-018

May 3, 2011

ABSTRACT

As systems become more software intensive and complex, managing their development and implementation also becomes more complex. Models in development today are isolated, domain-specific artifacts that are created throughout the design lifecycle. A mechanism is needed to integrate the design models with simulation environments as the models are being developed and refined in order to rapidly see the impacts as the design matures. The ability to perform verification, validation, and accreditation (VV&A) early in the modeling process and throughout the lifecycle could greatly improve the model and its contribution. But in order to perform VV&A on complex systems, a precise language would be required to model these systems in an integrated fashion to remove ambiguity and the segmented developmental lifecycle.

Objectives of this research included exploring the unique capabilities of Architectural Analysis and Design Language (AADL) for developing high confidence (verified and validated) models as part of a system development lifecycle and to determine the maturity of the AADL tools for VV&A model refinement. To show how AADL could be used to embed the Verification and Validation of architectural models into the development process, an architectural model of the Army's Systems Integration and Test Laboratory (STIL) was developed and used as a test bed.

The results of the research showed that a portion of a real world DoD representative system could be modeled using AADL in a very short time with little previous experience in AADL. AADL's well-defined semantics supported Architecture/System Design verification by allowing a precise specification of the architecture so that the analysis performed is trustworthy and repeatable.

This page intentionally left blank

TABLE OF CONTENTS

FIGURES AND TABLES

1 SUMMARY

As systems become more software intensive and complex, managing their development and implementation also becomes more complex. Modeling portions or all of a system is becoming more essential because of their contributions to design decisions early in the lifecycle that can impact cost, schedule, and performance. Models in development today are isolated, domain-specific artifacts that are created throughout the design lifecycle. A mechanism is needed to integrate the design models with simulation environments as the models are being developed and refined in order to rapidly see the impacts as the design matures. The ability to perform verification, validation, and accreditation (VV&A) early in the modeling process and throughout the lifecycle could improve the model and its contribution greatly. But in order to perform VV&A on complex systems, a precise language would be required to model these systems in an integrated fashion to remove ambiguity and the segmented developmental lifecycle. Objectives of this research included exploring the unique capabilities of Architectural Analysis and Design Language (AADL) for developing high confidence (verified and validated) models as part of a system development lifecycle and to determine the maturity of the AADL tools for VV&A model refinement. To show how AADL could be used to embed the Verification and Validation of architectural models into the development process, an architectural model of the Army's Systems Integration and Test Laboratory (STIL) was developed and used as a test bed.

The results of the research showed that a portion of a real world DoD representative system could be modeled using AADL in a very short time with little previous experience in AADL. AADL's well-defined semantics supported Architecture/System Design verification by allowing a precise specification of the architecture so that the analysis performed is trustworthy and repeatable. The ability to use custom property sets within AADL allowed the embedding of traceability information into the model. The traceability data provided a mechanism verify of values contained in the model to assist in the reusability of the model and overall verification of the model.

Possible benefits of using AADL include the ability to perform incremental VV&A on system architectures, perform trade studies on crucial components of the system and to discern deep architectural issues. AADL enables the detection of design or requirement problems related to the integration of the system and system level qualities while considering the impact on system reliability and safety.

2 INTRODUCTION

This report presents the research and findings for the use of Architecture Analysis and Design Language (AADL) and tools designed with AADL for the verification, validation and accreditation of system architectural models. The University of Alabama in Huntsville's (UAHuntsville) Rotorcraft Systems Engineering and Simulations Center (RSESC) explored the unique capabilities AADL offers for developing high confidence (verified and validated) models as part of a system development lifecycle. Specifically, we investigated automated verification through model consistency, semantic checking and traceability between the requirements. A conceptual model and a runtime model were created to perform constraint checking and model refinement in an attempt to show how AADL can improve the ability to verify and validate a system or simulation and to provide evidence that the accreditation process can be shortened. In order to explore the potential of AADL, a Department of Defense (DoD) system presently under development was selected as pilot project or test candidate. This DoD system, the System Test and Integration Lab (STIL), was modeled to understand and demonstrate AADL's capability to assist in incremental Verification, Validation and Accreditation (VV&A). Due to the limited scope of funding and time duration, the research was focused on the Time, Space, Position Information (TSPI) for the STIL. Under this task we also leveraged and expanded existing System Architecture Virtual Integration (SAVI) and AADL research in the area of incrementally verifying and validating using the AADL language.

Funding for this research task was also provided by the Project Manager of Instrumentation, Targets and Threat Simulators under the Program Executive Office for Simulation, Training and Instrumentation, (PEO-STRI/PM-ITTS) and informational support was provided by the Aviation Flight Test Directorate. This research work was done in partnership with the Aerospace Vehicle Systems Institute (AVSI) Consortium and the AADL's users group who participated in many of the briefings to the sponsor.

2.1 BACKGROUND

The RSESC at the UAHuntsville has been working with many of the Army's Redstone Arsenal offices and several other Army and Navy offices for many years assisting with systems engineering and system design problems. Over the last six months, RSESC has been working with the Army's Aviation Engineering Directorate (AED) and Software Engineering Directorate (SED) under the U.S. Army's Aviation and Missile Research, Development Engineering Center (AMRDEC) on how to manage complex systems including verifying and validating these systems. It is believed that a transition is required from document-centric requirements to integrated mathematical models to verify and validate those requirements.

Presently, there is not a way to predict the performance of complex systems all the way through integration. Software and system design languages are loosely defined and therefore do not provide the precise definition needed for high fidelity simulation and quantitative modeling and formal methods. When considering analysis tools there is a limitation in their capability to work together; therefore problems are typically found after the systems are built. It is believed that an architectural context is needed to resolve this issue. A high-level architecture specification allows end-to-end analysis or simulation of the complete system to help ensure system success before system integration and test.

Models must be developed with a common understanding of the semantics of the modeling elements in order to allow the integration of architectural models. Not having precise semantics and a common understanding of those semantics results in difficulty during the verification process due to the lack of precise analysis capability. The utilization of a custom language leads to a need to define and document it well. Upkeep and revalidation of assumptions become an issue each time models that have their own semantics are integrated.[1]

As part of this effort to resolve the shortfalls of the current analysis tools used for VV&A, research has been done to determine the best tools for verification, validation and accreditation of architectural models. Presently AED and SED are investigating the use of AADL as a formal architecture language in coordination with the SAVI consortium. In partnership with AED and SED, RSESC has attended courses and seminars on the topic. RSESC has also been actively involved in developing expertise in the AADL specification, model based engineering and toolsets that would be used in coordination with the AADL model.

For this research task RSESC leveraged other research work currently on contract with the U.S. Army. These efforts include working with PEO-STRI on establishing the baseline requirements and architectural format for Block II of the Redstone Test Center's System Test and Integration Lab.

2.1.1 System Architecture Virtual Integration Background

SAVI is an international industry consortium developing a new capability for early verification and validation of architecture supporting acquisition and development. SAVI participants are Boeing, Airbus, Lockheed Martin, British Aerospace Engineering Systems, Rockwell, Goodrich, Federal Aviation Administration, Department of Defense (DoD), and Carnegie Mellon (CMU) Software Engineering Institute (SEI).

The SAVI is a five year endeavor that is presently in its second year of research. It is a research effort to define the standards and technologies needed to effect virtual integration. The project is intended to be a global collaboration to integrate three emerging technologies: Model-based, Proof-Based, and Component-Based engineering.

One of SAVI's goals is to determine methods for structured/transformable data interfaces to change the acquisition paradigm to facilitate systems integration. SAVI is not a software tool or a design tool or a continuation of current system development practices.

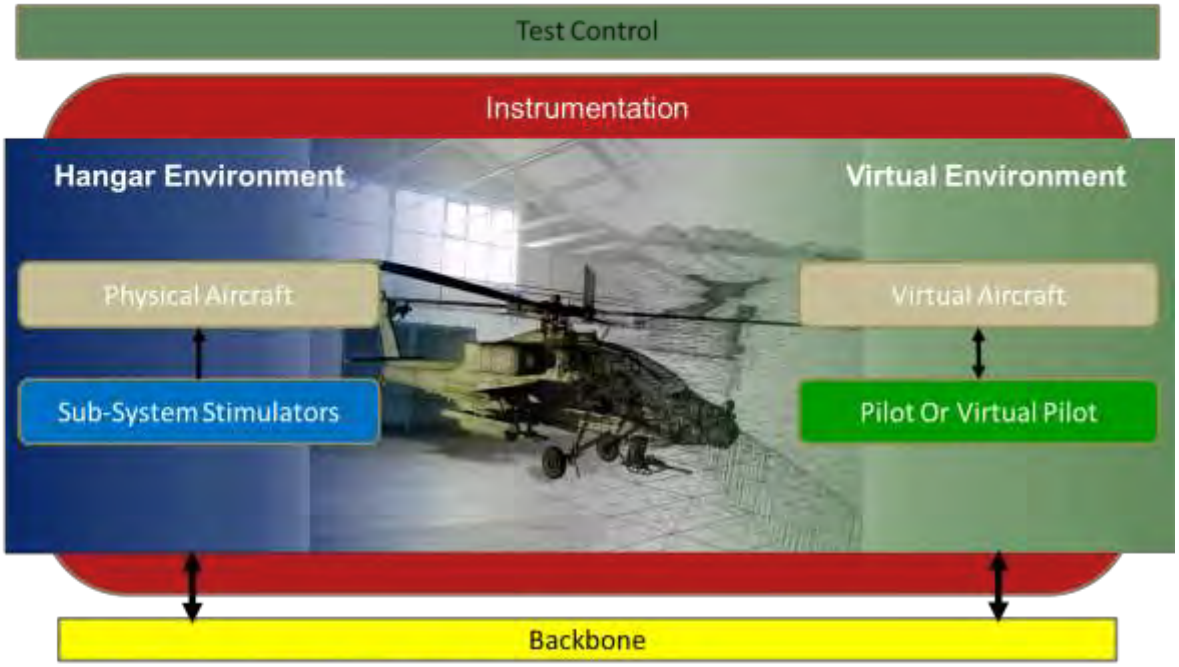
SAVI is investigating extending the semantic architecture model using the extensibility mechanism of AADL to support the capture and validation of requirements. "The end-to-end validation of systems involves validation of requirements against system models and system implementation. AADL properties support basic traceability between a requirements document and models, as well as traceability from models to implementation in the form of detailed design models and source code."[2] Additional detail on the System Architecture Virtual Integration and the work that has been done can be provided upon request.

Participants in the SAVI project were and continue to be open to collaboration with RSESC in the research being performed on AADL toolsets. RSESC's experience in rotorcraft, space and ground support systems is of benefit to the missions and goals for VV&A. Evaluating these tools and developing expertise in AADL specification provides UAHuntsville with state of the art capability to impact system integration and verification and validation.

2.1.2 System Test Integration Lab Overview

Digital transformation has resulted in a new generation of complex aircraft, constituting a new type of airborne "System of Systems." These new aircraft require a robust hardware and software environment for intelligent test and control that can efficiently test these aircraft in a real-time integrated system environment. This requires an analytical/intelligent philosophy for the evaluation of sophisticated complex systems. The STIL is intended to be an installed system test facility that will provide a synthetic environment capable of immersing an instrumented aircraft and its system in a controlled, repeatable and distributed virtual environment to enhance test capability; augment open-air testing; mitigate program risk, cost and schedule; and provide a collaborative environment for system of systems testing.[3] This vision of the STIL is presented in Figure 1, and actual photos of the facility from a January 2011 tour follow.

The STIL was selected for the research because it provided a real-world DoD system integration challenge. The STIL is a software-intensive distributed system that must produce precise and deterministic event ordering to meet requirements. It is a multifaceted system involving simulation and stimulation allowing for the ability to expand the AADL model into more areas/disciplines. Lastly, the AADL model of the STIL allows the analysis of various characteristics of the STIL architecture.



Slide Provided by PEO-STRI / PM-ITTS

Figure 1: STIL Concept Model[3]

Figure 2: CH-47 Aircraft a Future Aircraft to be Tested in the STIL

Figure 3: Picture inside of the CH-47 Aircraft within the STIL Hanger

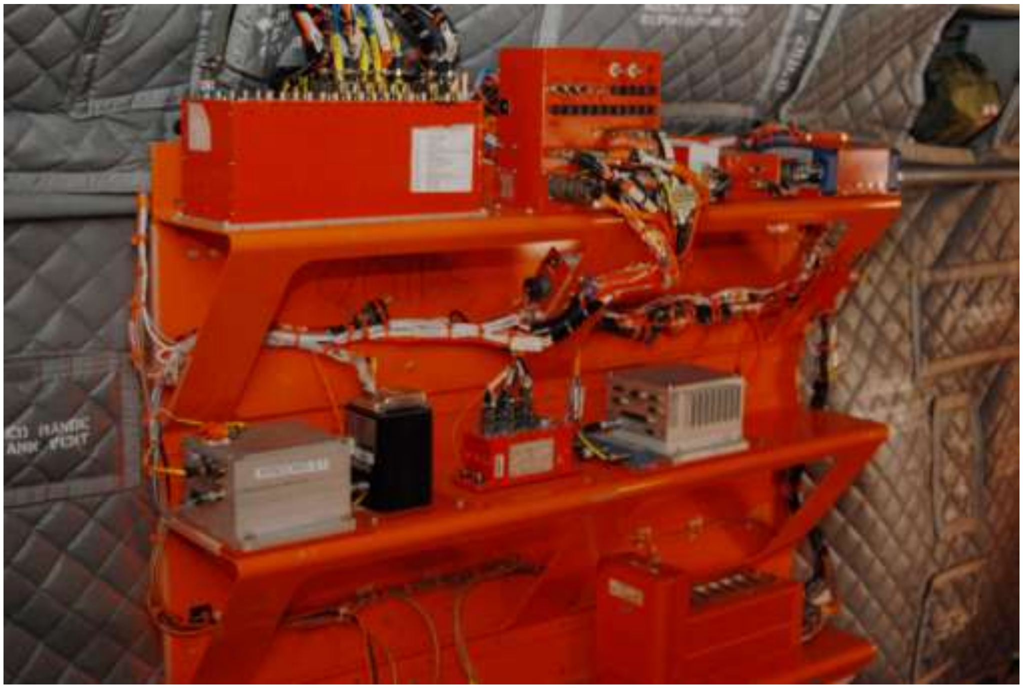


Figure 4: Picture of the STIL Stimulation Hardware

2.2 CURRENT PRACTICE

Presently, complex systems are becoming more reliant on software and embedded real time controllers. Modeling portions or all of a system is highly useful because it can contribute early in the lifecycle to design decisions that impact cost, schedule, and performance. Models in development today are isolated, domain-specific artifacts that are created throughout the design lifecycle. Models that contribute to key design decisions early in the lifecycle can have significant impacts on costs, schedule and performance. Currently, there is no true mechanism to integrate or refine the models and the simulations so that the impact to the system can be seen as its design matures. The ability to perform VV&A early in the modeling process and throughout the lifecycle can improve the model and its contribution greatly. Having a precise language to model systems in an integrated fashion is imperative to removing ambiguity and the segmented developmental lifecycle.

3 OBJECTIVE

The objective of this research was threefold:

- Explore unique capabilities of AADL for developing high confidence (verified and validated) models as part of a system development lifecycle.
- Create a test bed model of the STIL's TSPI data to understand and demonstrate how VV&A can be embedded into the development of architectural model using AADL.
- Determine the maturity of the AADL tools for VV&A model refinement.

AADL can support VV&A by using a standardized foundation for the Modeling and Simulation (M&S) of a system. This research is the beginning step in showing how verification and validation time can be reduced. Verification is supported by the AADL syntax checker and analysis tools that can by automated to find known potential problems with the model. Model validity is supported by the ability to use the same model as an analysis model to predict system performance and as a specification of the design used during system implementation. The implementation of the system should match the analysis model if it is built to the specifications.

The deliverables from this research were the following:

1. A test bed built and provided to explore how development times could be reduced by integrating verification and validation into the model development process by using the AADL
2. A demonstration of the impacts and benefits of using AADL in the verification and validation process including the ability to use the tools on DoD designs and determining the ability to migrate toolsets to the user community (learning curve, challenges, etc.)

4 METHODOLOGY

AADL was chosen for this research task because AADL provides standardized, well documented semantics to underlay tools, enabling ease of integration of tools based on AADL. The language has strong semantics and textual notation that enables integration of architecture specifications across entities. When working with other languages the loose or weak semantics require each user to add semantics which can be a major source of errors, inconsistencies, and undefined assumptions. AADL's well-formed architectural behavior semantics cover nominal and fault management behavior, and allow assessment of faulty or incomplete models. The toolsets have the ability to provide incremental verification and validation leading to a qualified system. At each stage, starting with the requirements, we can validate that the requirements (and then their allocated constraints on the next level) are sufficient and verify the correctness of the model at that level. The ultimate result through all phases of verification is a qualified system.[1]

To show how AADL could be used to embed the Verification and Validation of architectural models into the development process, an architectural model of the STIL was developed and used as a test bed. The following process was followed:

1. Develop a conceptual architectural model that includes the logical functional blocks of the architecture, relevant performance requirements, and traceability to the system specification.
2. Verify the conceptual architectural model by using automated tools to verify model semantics and completeness.
3. Develop a runtime architectural model that includes the software and execution platform components of the system, relevant properties for analysis, and traceability to the conceptual architecture model and to the sources used to derive values in the model.
4. Verify the runtime architectural model by using automated tools to verify model semantics, completeness, and whether the modeled architecture fulfills performance requirements specified in the conceptual architecture.
5. Validate analyses performed on the architectural models.
6. Continually engage the stakeholders in an iterative manner to refine the research.

Figure 5 shows the how traceability was embedded in the model using a requirement from the STIL System Specification, ASY-2526, as an example.

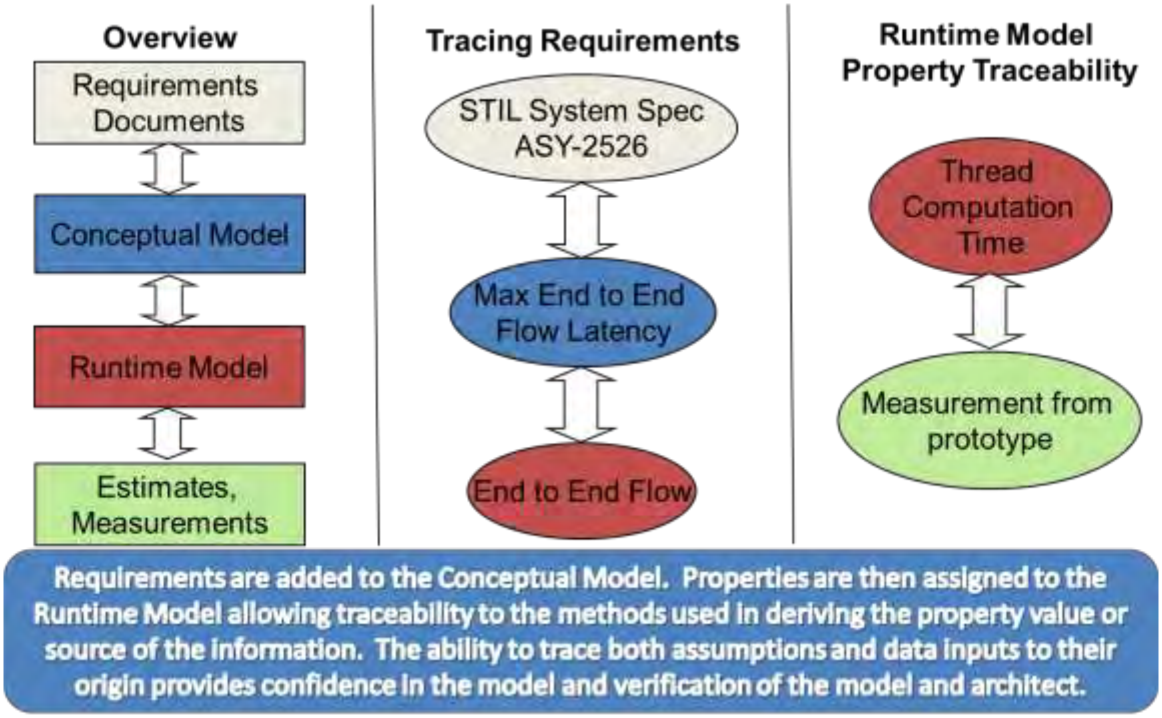

Overview
Requirements Documents
Conceptual Model
Runtime Model
Estimates, Measurements
Tracing Requirements
STIL System Spec ASY-2526
Max End to End Flow Latency
End to End Flow
Runtime Model Property Traceability
Thread Computation Time
Measurement from prototype
Requirements are added to the Conceptual Model. Properties are then assigned to the Runtime Model allowing traceability to the methods used in deriving the property value or source of the information. The ability to trace both assumptions and data inputs to their origin provides confidence in the model and verification of the model and architect.

Figure 5: Traceability Process Overview

5.0 Research and Results

The architectural model of the STIL was developed using the AADL. The Society of Automotive Engineers (SAE) standardizes AADL and there exists an assortment of tools available to work with it. Version 1 of the AADL was used during this task. The following tools were used for this task:

- OSATE – An Open-Source AADL development environment developed by the SEI.
- Ocarina – A tool suite for working with AADL models developed by Télécom ParisTech.

5.1 Development of the Conceptual Architecture Model

The modeling process began with the creation of a conceptual architecture model. The conceptual architecture model contained the high-level functional components of the STIL. It also contained a specification of the end-to-end data flow through the functional components. Due to time and funding limitations, the focus of the modeling was on the elements that were of highest priority in the design of the STIL. The element of focus was related to the TSPI. A graphical representation of the conceptual architecture model is presented in Figure 6.

5.1.1 Conceptual Elements

In order to model the conceptual architecture in the AADL, a custom property was used to designate an element as being part of the conceptual architecture. This designation was needed in order to differentiate between elements of the runtime and the conceptual architecture during automated analysis. The custom property was required because version 1 of the AADL does not support this capability. Version 2 of the AADL features an "abstract" component type that addresses this shortfall and may be used to model conceptual architectures instead of using a custom property.

5.1.2 Property Traceability

The conceptual architecture model contains performance requirements specified in the system specification to allow for the analysis of the modeled architecture. In order to aid in model verification, the performance requirements need to be traceable to the system specification. An example of a performance requirement is the maximum latency of the end-to-end flow for the TSPI. The latency property in AADL was used to specify the maximum allowed latency. In order to embed the mapping from the latency value to system specification a new property was introduced. The property allows the modeler to embed references to the requirements in the system specification that were used to derive the latency value. To make the concept generic, the idea of a "shadow property set" was introduced. The shadow property set contains properties that complement the properties in the main property set. An example of a partial shadow property set is contained in Appendix B. Shadow property sets and their properties followed a consistent naming scheme designed to allow usage with existing property sets and automated analysis.

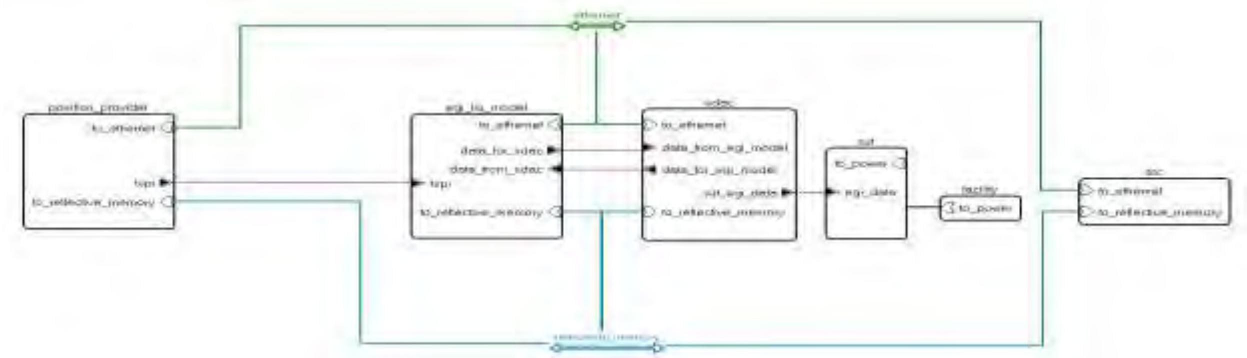


Figure 6 Graphical Representation of the Conceptual Architecture Model

5.2 Verification of the Conceptual Architecture Model

The conceptual architecture model needed to be verified to provide confidence that it contained the necessary information for the desired analysis of the TSPI data flow.

The AADL specifies rules for the legality of units and property assignments based on the property definitions. AADL tool suites such as OSATE and Ocarina automatically check the model to ensure these rules are being met. This helps verify the model does not violate the semantics of the language.

Ocarina contains an implementation of a constraint language named the Requirement Enforcement Analysis Language (REAL) as an annex. REAL allows an AADL modeler to specify constraints that can be checked by Ocarina. REAL was used to help verify the conceptual architecture model by adding constraints to check if performance requirements were specified on relevant elements and that those requirements were traceable to the system specification. The check was done by adding REAL constraints that verified the existence of property values in the AADL model. In general, constraint languages such as REAL allow specification of constraints that can verify the model meets custom modeling rules and to check for known potential problems. In many cases, these constraints can be shared between multiple models. Constraints are especially useful with the handling of custom property sets; they allow verification of proper usage of the custom properties.

A custom OSATE analysis plugin was used to generate a requirements traceability report from the conceptual architecture model. The report uses the requirement traceability properties to generate a report which shows the traceability between the requirements contained in the system specification and the properties in the conceptual architectural model. An example portion of a generated report can be found in Appendix D.

5.3 Development of the Runtime Architecture Model

In order to analyze the flow of the TSPI through the system, a model of the runtime architecture model was developed. The runtime architecture model was based on the initial Block 1 implementation of the STIL. While the components of the conceptual architecture model corresponded with logical functional components of the system, components of the runtime architecture model correspond to software and hardware components of the system implementation. Some of the types of components it contains are:

- Processes
- Processors
- Threads
- End to end flow specifications
- Buses
- Protocols

Although the model is based on the Block 1 implementation of the STIL, for logistical reasons, the execution platform components such as processors and buses are not reflective of the STIL block 1 implementation. Rather, it is representative of a possible hardware configuration of the system. Two versions of the runtime architecture were modeled. One version utilized an Ethernet bus for communication and the other utilized a reflective memory bus. Both versions share a common foundation using the refinement mechanism in AADL. The hardware configuration is shown in Figure 7 where "eth" is the abbreviation for Ethernet, and "rm" is the abbreviation for reflective memory.

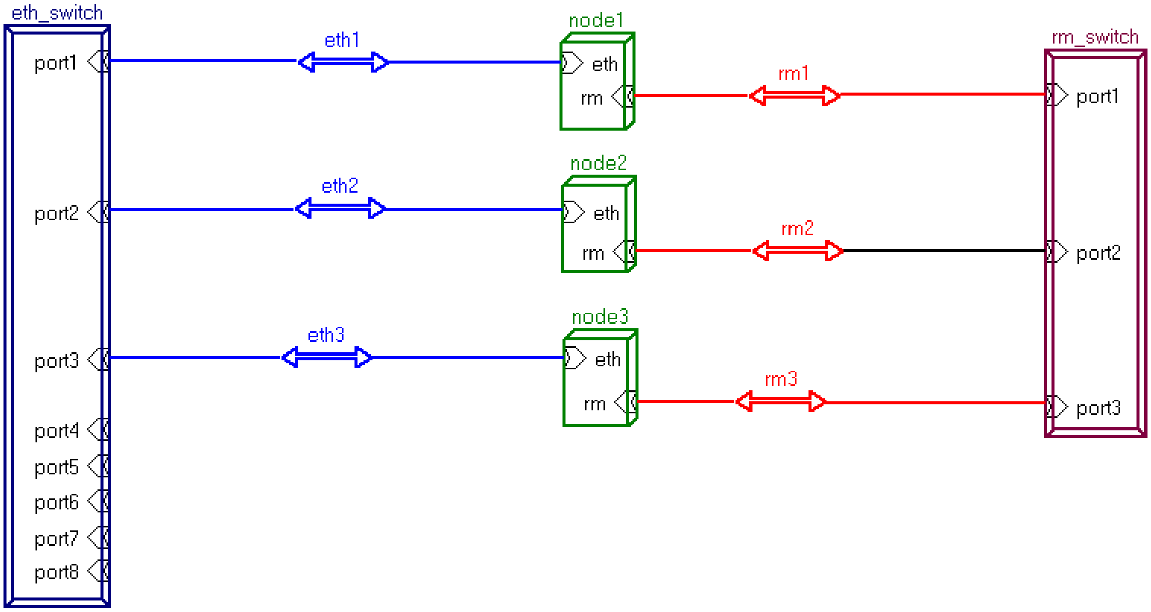
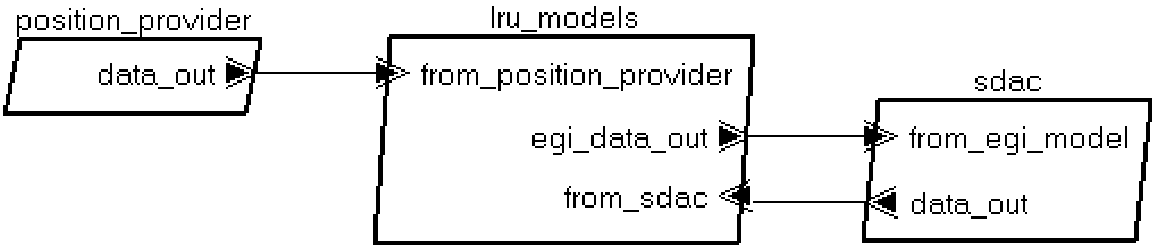


Figure 7 Hardware portion of the runtime architecture model

The software components of the model directly map to components in the implementation of the STIL. Figure 5 shows the three processes involved with the generation and transfer of TSPI data to the System Under Test (SUT).



lru – Line Replaceable Unit
sdac – System Data Acquisition
egi - Embedded GPS/INS (Global Positioning System/Inertial Navigation System)

Figure 8 Top level of the software portion of the runtime architecture model

5.3.1 Traceability to the Conceptual Architecture Model

Elements of the runtime model needed to be traceable to the conceptual model to allow traceability from the system specification to the runtime model and to allow performance requirements found in the conceptual architecture model to be accessible when analyzing the runtime architecture model. The traceability aids in verifying that the runtime architecture model reflects the conceptual architecture. The traceability was embedded into the model by using custom properties to map elements in the runtime model to elements in the conceptual architecture model. The custom property set is contained in Appendix C. Another method of mapping between the conceptual architecture to the runtime architecture is the refinement mechanism found in the AADL. Both of these techniques are described in version 2 of the AADL standard. Properties were used to specify the mapping to allow for added flexibility by allowing a single runtime component to potentially be traceable to multiple conceptual components.

5.3.2 Tracing Property Values to Origin

When modeling a system, values entered as part of the model can derived in a number of different ways, e.g. estimation, calculation, and measurements. In AADL these values become property values. The method used to obtain a property value is important in order to gain confidence in the model. Custom properties were used to denote the method used to obtain the property values contained in the runtime architecture model. The property values were contained in the same shadow property sets used to trace performance requirements to the system specification. An example property set can be found in Appendix B. Embedding the derivation method into the model promotes model reuse because other users of the model have information needed to ensure that the model is appropriate for the intended use.

5.4 Verification of the Runtime Architecture Model

Various automated analyses were performed on the runtime architecture model in order to verify that it contained needed information and met the performance requirements specified in the conceptual architectural model. The precise semantics of AADL supports automated verification efforts by allowing multiple tools to work on a single model based on a common understanding. An OSATE plug-in was used to check the consistency of the binding of port connections to the execution platform components. The plug-in was used to verify that the connections specified in the model were semantically valid. REAL was used to ensure the existence of properties for the tracing to the conceptual architecture model and the derivation method of property values. It was also used to ensure the existence of properties needed for other analysis and to verify that all threads were bound to a processor.

Several reports were generated from the architectural model as a method of generating artifacts showing the traceability built into the model. This could also be beneficial when looking at accrediting a system. The *Conceptual to Runtime Traceability Report* shows the relationship between the runtime and the conceptual architecture models. The *Derivation Method Report* contains properties found in the runtime architecture model and the method used to derive their values. Example portions of the reports can be found in Appendix D.

Communication analysis was performed in order to determine the performance characteristics of the modeled system. The automated analyses used the runtime architectural model to calculate the bandwidth usage and the latency of the end-to-end-flows specified in the system. This calculation was done in a custom OSATE analysis plugin. The end-to-end flow latency was calculated using the method described in the SEI Technical Note, *Flow Latency Analysis with the Architecture Analysis and Design Language (AADL)*. The analysis utilized the traceability from the runtime architecture model to the conceptual architecture model to retrieve performance requirements. The analysis tool helps to verify that the design of the system meets the requirements specified in the conceptual architecture by comparing the required maximum latency with the calculated maximum latency. The analysis can be applied iteratively. As more detail is added to the architectural model, the analysis generates more precise results therefore illustrating incremental verification and validation. The analysis generated a bandwidth and latency report. Examples of the reports can be found in Appendix D.

5.5 Validate Analyses

Models and analysis must be validated in order to ensure that results reflect the modeled system to enough of a degree to be used for the desired purpose.

A code generation technique was used to demonstrate how an analysis and architectural runtime model could be validated early in the development process. The latency portion of the communication was used for the demonstration. To generate data to compare against, a code generator was used to generate C code based on the runtime architectural model. A program was generated for each process specified in the runtime architecture model. The generated code was executed in a lab setup that reflected the hardware described in the runtime architecture model. The generated programs exchanged data at the correct rates and used the data structures described in the architecture model. The programs also generated log files that were post-processed in order to generate a report containing the calculated latency of the end-to-end flows in the running system. The generated report is shown in Appendix D. The results of the validation effort showed that in 3 out of the 4 runs, the measured values were within the expected range. In one run, the maximum latency was greater than the calculated worst-case latency. It is believed it was caused by the lack of a precise time synchronization technique. Further analysis would be required to come to a conclusion of the root cause.

5.6 Briefings to the Stakeholders

Throughout the research task RSESC continually engaged the stakeholders in an information and feedback exchange to refine the approach and to ensure it was meeting the goals of all the stakeholders. RSESC worked to bridge the gap between fundamental and applied research in order to transfer this knowledge to the user community. It was a goal to ensure that the results from this task not only benefited the VV&A community but also the user community. The meetings were done both formally and informally with PEO-STRI/PM-ITTS, the Army's Flight Test Directorate under the Redstone Test Center, AMRDEC's AED, AMRDEC's SED, the AADL User's group leaders and SAVI leaders.

The final out briefing for the VV&A task occurred in Huntsville, Alabama on January 20, 2011. It was a two day event and began with Phil Zimmerman briefing at the American Helicopter Society's Huntsville Chapter monthly technical luncheon meeting about the VV&A initiatives. After the luncheon, UAHuntsville RSESC researchers and the sponsors of UAHuntsville's research task, PEO-STRI/ PM-ITTS and Office of the Secretary of Defense, toured the Redstone Test Center's STIL facility.

The attendees list and agenda for the visit is provided in Appendix E.

6 Overall Research Findings

A primary objective was to explore the unique capabilities of AADL for developing high confidence (verified and validated) models as part of a system development lifecycle. To meet this objective, RSESC created a test bed model of the STIL's TSPI data to understand and demonstrate AADL's capability in regards to incremental VV&A.

The research shows that the AADL language forces the model to conform to a set of rules which help verify the model. Having a standardized model of the system allows analyzing the model to find potential problems. AADL's extensible nature is useful for enhancing overall capabilities and aids in the verification and validation of architectural models created in it. Custom annexes allow sublanguages, such as REAL, to be embedded within the model for verification purposes. AADL's well-defined semantics do support Architecture/System Design verification by allowing a precise specification of the architecture so that the analysis performed on it is trustworthy and repeatable. AADL's ability to use custom property sets allows the embedding of traceability information into the model. The traceability data then aids in the verification of values contained in the model.

This objective was met by successfully modeling the flow of the TSPI data in the STIL as well as modeling Ethernet and reflective memory variations of the system architecture.

By modeling the STIL, clear evidence was found as to the value of using a model based engineering languages such as AADL.

It was previously thought that the primary contributor to latency and jitter on the STIL was the bus used for communication. RSESC was able to determine through that in the case of the STIL, the usage of globally asynchronous threads along with data sampling was the primary contributor to latency and jitter. This analysis was done through incrementally verifying and validating the model. Further, it was determined that the lack of synchronization and the usage of data sampling cause significant latency because threads will not consume new data until the thread period after the data is produced.

AADL did prove to have multiple strengths. First, there are significant amount of resources concerning AADL on the internet that include examples, wiki, and scholarly papers. Secondly, copies of the AADL standard are available from the SAE. And lastly, it is reasonably easy to learn for someone with knowledge of software systems.

On the other side were the weaknesses. The toolsets had some maturity issues in several different areas. At the time of the research effort, the current stable version of OSATE does not support AADL V2 and lacked a mature graphical editor. Ocarina does support AADL V2 but with limitations. There were minor issues concerning AADL terms which are slightly different than existing terms. The usage was understandable and defined in the AADL standard, though time was required to learn nuances. But by using version 1 of AADL many of these weaknesses were overcome and good results were found for the overall STIL design.

7 CONCLUSION

RSESC was able to illustrate the usability of AADL in a very short research task. It was shown that a portion of a real world DoD representative system could be modeled using AADL in a very short time with little knowledge of AADL. Valuable feedback was given to PEO-STRI in the development of the STIL in regards to latency of the Ethernet verses reflective memory of the system.

UAHuntsville RSESC completed all of the following tasks in less than six months:

- Clearly defined a task that supported the effort
- Selected a system to model
- Worked in coordination with the AADL Users Group, SAVI and PEO-STRI
- Designed and developed the model

- Applied real world DoD system specifications to the model
- Learned AADL and its caveats

- Refined knowledge of VV&A and M&S
- Wrote open source code to address software maturity issues
- Came up to speed and used OSATE and Ocarina analysis tools as well as developed other analysis tools to complete the analysis
- Validated the results of the analysis by generating code from the model and running it

From this research the benefits of using AADL were seen including the ability to perform incremental VV&A on system architectures, perform trade studies on crucial components of the system and to discern deep architectural issues. AADL can detect design or requirement problems related to the integration of the system and system level qualities while considering the impact on system reliability and safety.

Long term benefits include the ability to reduce overall costs as found in the SAVI costs benefits study. The first aircraft program to apply SAVI saved in the worst case $700 million up to the best case of $2 billion. Expanding the same philosophy using AADL in the M&S community could help save additional money. And lastly, expanding the research to partner with the US Army's Research, Development and Engineering Command (RDECOM), AMRDEC, System-Simulation-and -Development-Directorate facilities or the Joint Technology Center/Systems Integration Laboratory to explore how or if AADL could be further utilized to fill VV&A gaps for simulations.

When looking at the benefits to the stakeholders, many benefits can be foreseen. Developers of complex software-reliant systems could lower development costs by building more effective systems thus saving money to the taxpayer. Modeling systems in a precise design language could allow the DoD the ability to rapidly build and add flexibility to its systems, which in turn would maintain the leadership role it enjoys in fields like aerospace. This technique in turn could be shared with other fields that use M&S for safety critical and complex systems as well as with the acquisition community and stakeholders involved with M&S and system development. Successful research would allow for new methods and knowledge on incremental VV&A and how it might integrate into an overall process including virtual integration and enhanced use of qualified SILs and STILs. This research expands on existing SAVI and AADL research in the area of incremental verification and validation using the AADL language to demonstrate benefits for future and existing systems. Lastly furthering this concept on real DoD systems, could provide a foundation of new methods and ideas for the M&S community, the system development community, Project Managers and the students, our future designers and developers.

8 RECOMMENDATIONS

For this task interested parties from several disciplines that see value in the overall modeling of a system were brought together. Participation in this research task came from several communities, the flight test community, system developers, users, simulation community, the Army's AMRDEC AED that is in charge of flight worthiness and the Army's AMRDEC SED. As future phases for this task unfold, the realization of breaking down the barriers to use one model or slight variations of a similar model to perform multiple tasks for system development should be viable.

One recommendation for a follow on to this research would begin to look at the possibility of overlaying planned TCD efforts of a program with a Model - Based Engineering (MBE) process, including three stages which correspond to three S&T acquisitions:

1. Overlay Configuration Trades and Analysis with Conceptual Model Development.
2. Overlay Air Vehicle Development with M&S Development, including constructive simulations and reconfigurable virtual prototypes.
3. Overlay Mission System Development with Model Instances in System/Software Integration Labs (SIL).

Using this approach for programs will make the requirements more consistent and traceable, and will allow for more exploration of design solutions using the model and correlated simulation tools. Having such a precise model will link the architectures of the system, the simulations, and the SILs for rapid airworthy code development and qualification which will have a significant impact in reducing development costs. As a result, the effort will shift into designing adaptable systems rather than evaluating point solutions. Using MBE will also allow the system design to be built upon from many different areas of expertise and result in a singular truth representation - thereby avoiding sole reliance on competing and often incompatible proprietary design documents. MBE will further break down stovepipes between the respective communities of design, performance evaluation, and qualification.[4]

Continue expanding the STIL model to further demonstrate benefits and weakness:

- Model potential connections between the Virtual Prototyping Model and the Redstone Test Center (RTC) STIL to demonstrate System of Systems integrated flight of Joint Multi Role (JMR) with current platforms and enable transition of virtual designs to operational software.
- Demonstrate the ability to expand the existing model to system testers and merge models from other sources while performing incremental VV&A.
- Integrate the STIL model with the AADL helicopter model that is in development by SEI and the AADL users group to demonstrate end-to-end modeling capability.

- Compare the communication analysis results with Block 1 STIL communication characteristics to validate the model and the analysis.
- Presently working with Wayne State University and TARDEC on Platform- Based Engineering and Model-Based Engineering to expand and explore possible benefits to Ground PMs and Aviation PMs. .
- Questions to consider:
  - Can AADL help in building a system that allows flexibility, versatility while preserving V&V already performed on the system or model?
  - How can AADL help in modeling legacy systems?
  - Are there techniques (Methods, Processes and Tools) that are being utilized by the JMR project that could be beneficial to ground systems?

Virtual Prototyping of JMR Army Rotorcraft or the Future Vertical Lift Initiative

- Use virtual prototyping as a means to reduce risk in design of JMR. (Combined submission from SSDD, SED, AED, and UAHuntsville)
- Use AADL to develop a conceptual model of JMR (UAHuntsville effort in conjunction with AED and SED)
- Implement the model in the development of a cockpit simulator in the SSDD APEX 2 lab, incorporating the AED flight model.
- Link virtual cockpit through distributed simulation to the SED SIL/ASIF.

In summary, based on the positive results found during this research, a number of future research areas/tasks are available:

- Models as specs
- Model-based design documentation
- Platform based engineering
- Incremental and adaptable verification and validation of systems and simulations
- Auto-generation of test plans and reports
- Trusted evidence generated using tools/automation
- Expansion of the research into utilizing the same tool to perform incremental verification and validation of systems and simulations to leverage models, systems engineering processes and the V&V process

APPENDICES

APPENDIX A REFERENCES

1. The sections noted with a footnote 1 within the final report were written using the following reference papers and presentations and during informal discussions with the individuals listed below. This work is an extension of, and complimentary to, the on-going research tasks of the AADL users group and SAVI project.

   Referenced Papers and Presentations:

   a. **The SAE Architecture Analysis & Design Language (AADL) Standard,** Peter H. Feiler, Software Engineering Institute, January 2008.
   b. **Challenges in Validating Safety-Critical Embedded Systems,** Peter H. Feiler, Software Engineering Institute, Copyright © 2009 SAE International, 09ATC-0271.
   c. **Diagrams and Languages for Model-Based Software Engineering of Embedded Systems: UML** [Unified Modeling Language] **and AADL,** *Dionisio de Niz, Software Engineering Institute, Carnegie Mellon.*
   d. **Multi-Dimensional Model Based Engineering for Performance Critical Computer Systems Using the AADL,** B. Lewis, P. Feiler, Proceedings from Embedded Real Time Software and Systems (*ERTS*^{2}*), 25-27 January 2006, Toulouse.*
   e. **System Architecture Virtual Integration: A Case Study,** P. Feiler, L. Wrage, J. Hansson. Proceedings from Embedded Real Time Software and Systems (*ERTS*^{2}*) 2010.*
   f. **Flow Latency Analysis with the Architecture Analysis,** Peter H. Feiler, Jörgen Hansson, *Carnegie Mellon University Software Engineering Institute,* Technical Note, CMU/SEI-2007-TN-010, December 2007.

   Individual Contributions by:

   a. Dr. Bruce Lewis, US Army RDECOM, AMRDEC, Software Engineering Directorate.
   b. Dr. Dave Redman, Aerospace Vehicle Systems Institute (AVSI), Texas A&M University.
   c. Dr. Don Ward, Aerospace Vehicle Systems Institute (AVSI), SAVI Program, Texas A&M University.

2. The sections noted with a footnote 2 within the final report were written using the following reference papers and presentations provided by AVSI and during informal discussions with the following listed individuals.

   Referenced Papers and Presentation:

The SAE Architecture Analysis & Design Language (AADL) Standard, Peter H. Feiler, Software Engineering Institute, January 2008.

Individual Contributions by:

a. Dr. Bruce Lewis, US Army RDECOM, AMRDEC, Software Engineering Directorate.

b. Dr. Dave Redman, Aerospace Vehicle Systems Institute (AVSI), Texas A&M University.

c. Dr. Peter Feiler, Software Engineering Institute, Carnegie Mellon University.

d. Dr. Don Ward, Aerospace Vehicle Systems Institute (AVSI), SAVI Program, Texas A&M University.

3. The sections noted with a footnote 3 within the final report were written using the following reference papers and presentations provided by PEO-STRI/PM-ITTS and during informal discussions with by PEO-STRI/PM-ITTS and the Army RTC.

4. The sections noted with a footnote 4 within the final report were written referencing the Director, Defense Research and Engineering (DDRE) Systems 2020 RFI, entitled "Model-Based Engineering (MBE) of Joint Multi-Role (JMR) Aircraft", addressing Topic VII.

Authors:

Greg Tackett, SES
Director for System Simulation and Development,
US Army RDECOM, AMRDEC

Ned Chase
Chief, Platform Technology Division,
Aviation Applied Technology Directorate,
US Army RDECOM, AMRDEC
JMR TCD Project Manager

Scott Dennis
ASIF Manager
Software Engineering Directorate,
US Army RDECOM, AMRDEC

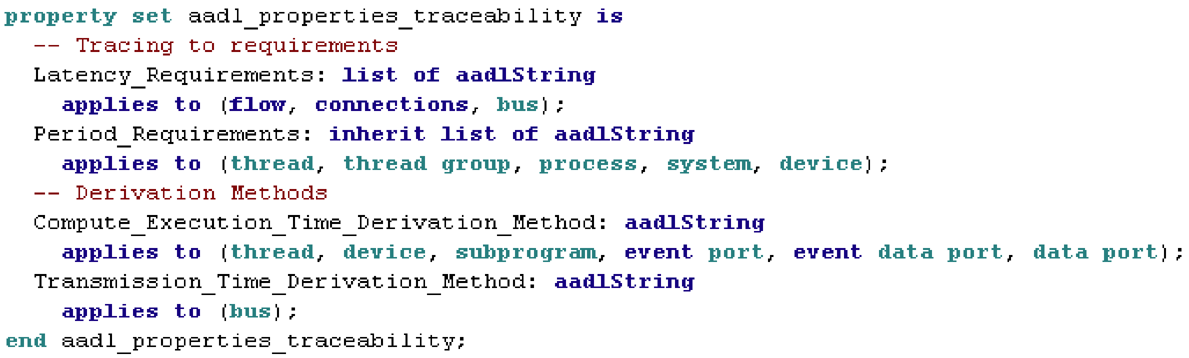
Robert L. King, PhD
Chief, Aeromechanics Division
Aviation Engineering Directorate,
US Army RDECOM, AMRDEC

Alex Boydston
Electronics Engineer, Mission Equipment Division
Avionics Branch
Air Vehicle Management Team
Aviation Engineering Directorate,
US Army RDECOM, AMRDEC

Bruce Lewis
Computer Engineer
Software Engineering Directorate,
US Army RDECOM, AMRDEC

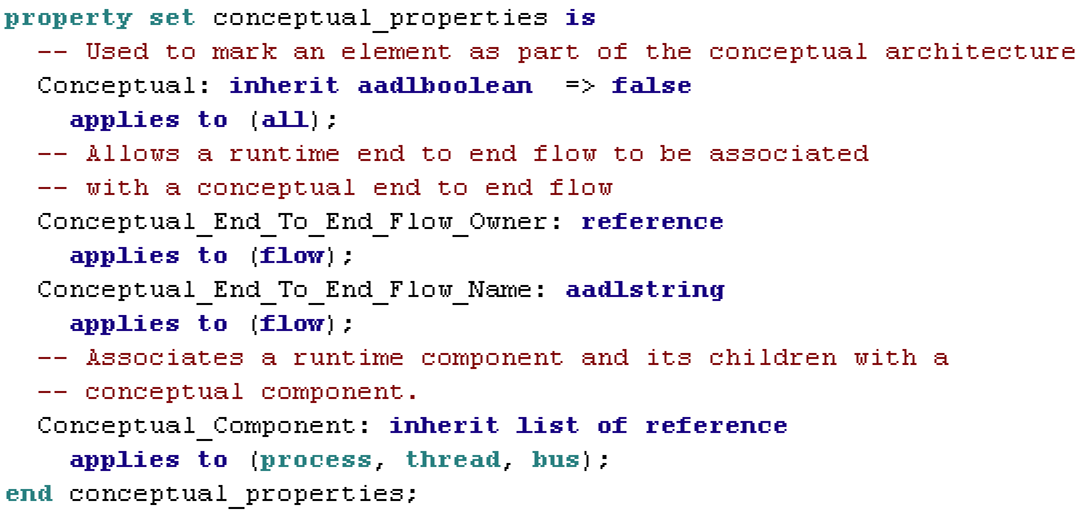
Sue O'Brien
Acting Director
Rotorcraft Systems Engineering and Simulation Center
University of Alabama in Huntsville

APPENDIX B SHADOW PROPERTY SET FOR AADL PROPERTIES



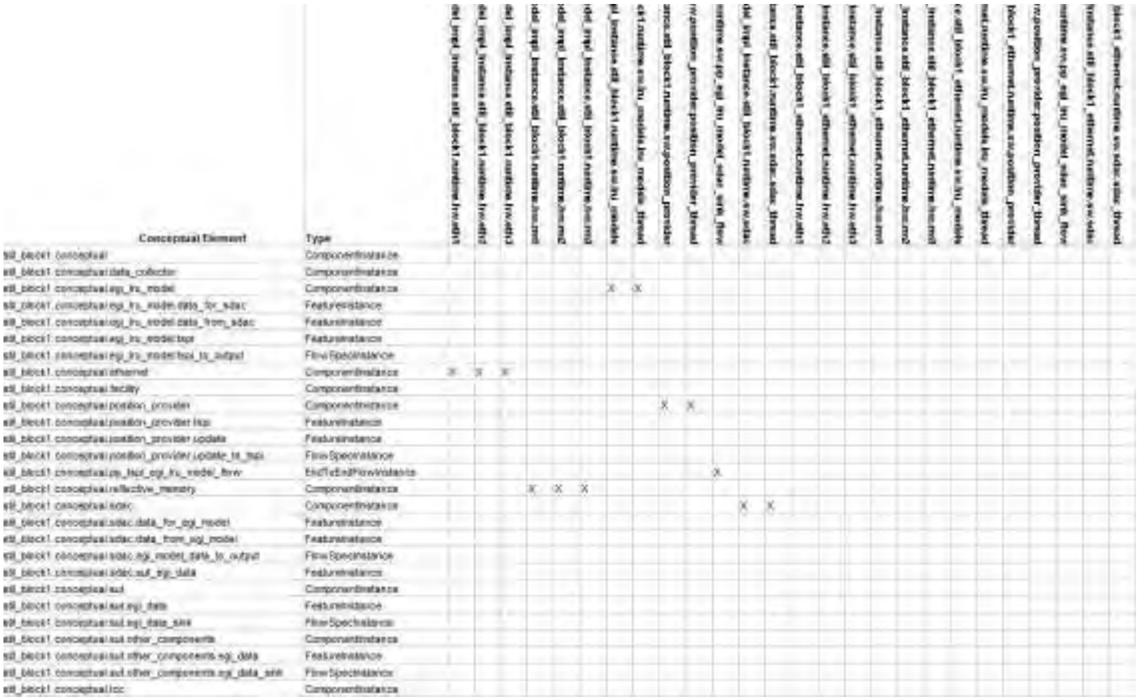
```
property set aadl_properties_traceability is
  -- Tracing to requirements
  Latency_Requirements: list of aadlString
    applies to (flow, connections, bus);
  Period_Requirements: inherit list of aadlString
    applies to (thread, thread group, process, system, device);
  -- Derivation Methods
  Compute_Execution_Time_Derivation_Method: aadlString
    applies to (thread, device, subprogram, event port, event data port, data port);
  Transmission_Time_Derivation_Method: aadlString
    applies to (bus);
end aadl_properties_traceability;
```

APPENDIX C CONCEPTUAL PROPERTIES



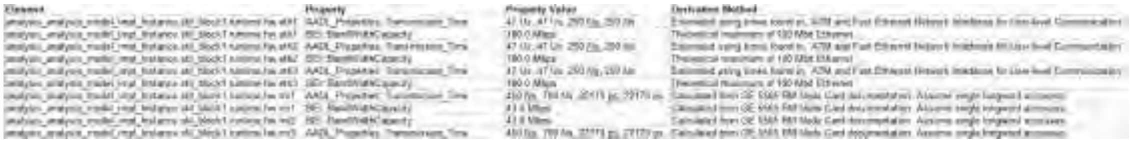
```
property set conceptual_properties is
  -- Used to mark an element as part of the conceptual architecture
  Conceptual: inherit aadlboolean  => false
    applies to (all);
  -- Allows a runtime end to end flow to be associated
  -- with a conceptual end to end flow
  Conceptual_End_To_End_Flow_Owner: reference
    applies to (flow);
  Conceptual_End_To_End_Flow_Name: aadlstring
    applies to (flow);
  -- Associates a runtime component and its children with a
  -- conceptual component.
  Conceptual_Component: inherit list of reference
    applies to (process, thread, bus);
end conceptual_properties;
```

APPENDIX D REPORTS



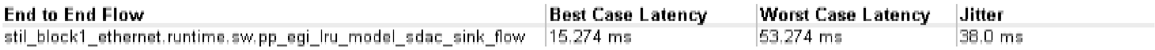
Conceptual Element	Type	del_impl_instance.stil_block1.runtime.hw.eth1	del_impl_instance.stil_block1.runtime.hw.eth2	del_impl_instance.stil_block1.runtime.hw.eth3	odel_impl_instance.stil_block1.runtime.hw.rm1	odel_impl_instance.stil_block1.runtime.hw.rm2	odel_impl_instance.stil_block1.runtime.hw.rm3	el_instance.stil_block1.runtime.sw.lru_models	ck1.runtime.sw.lru_models.lru_models_thread	ance.stil_block1.runtime.sw.position_provider	rv.position_provider.position_provider_thread	runtime.sw.pp_egi_lru_model_sdac_sink_flow	del_impl_instance.stil_block1.runtime.sw.sdac	ance.stil_block1.runtime.sw.sdac.sdac_thread	instance.stil_block1_ethernet.runtime.hw.eth1	instance.stil_block1_ethernet.runtime.hw.eth2	instance.stil_block1_ethernet.runtime.hw.eth3	instance.stil_block1_ethernet.runtime.hw.rm1	instance.stil_block1_ethernet.runtime.hw.rm2	instance.stil_block1_ethernet.runtime.hw.rm3	ce.stil_block1_ethernet.runtime.sw.lru_models	net.runtime.sw.lru_models.lru_models_thread	block1_ethernet.runtime.sw.position_provider	rv.position_provider.position_provider_thread	runtime.sw.pp_egi_lru_model_sdac_sink_flow	instance.stil_block1_ethernet.runtime.sw.sdac	block1_ethernet.runtime.sw.sdac.sdac_thread
stil_block1.conceptual	ComponentInstance																										
stil_block1.conceptual.data_collector	ComponentInstance																										
stil_block1.conceptual.egi_lru_model	ComponentInstance							X	X																		
stil_block1.conceptual.egi_lru_model.data_for_sdac	FeatureInstance																										
stil_block1.conceptual.egi_lru_model.data_from_sdac	FeatureInstance																										
stil_block1.conceptual.egi_lru_model.tspi	FeatureInstance																										
stil_block1.conceptual.egi_lru_model.tspi_to_output	FlowSpecInstance																										
stil_block1.conceptual.ethernet	ComponentInstance	X	X	X																							
stil_block1.conceptual.facility	ComponentInstance																										
stil_block1.conceptual.position_provider	ComponentInstance									X	X																
stil_block1.conceptual.position_provider.tspi	FeatureInstance																										
stil_block1.conceptual.position_provider.update	FeatureInstance																										
stil_block1.conceptual.position_provider.update_to_tspi	FlowSpecInstance																										
stil_block1.conceptual.pp_tspi_egi_lru_model_flow	EndToEndFlowInstance											X															
stil_block1.conceptual.reflective_memory	ComponentInstance				X	X	X																				
stil_block1.conceptual.sdac	ComponentInstance												X	X													
stil_block1.conceptual.sdac.data_for_egi_model	FeatureInstance																										
stil_block1.conceptual.sdac.data_from_egi_model	FeatureInstance																										
stil_block1.conceptual.sdac.egi_model_data_to_output	FlowSpecInstance																										
stil_block1.conceptual.sdac.sut_egi_data	FeatureInstance																										
stil_block1.conceptual.sut	ComponentInstance																										
stil_block1.conceptual.sut.egi_data	FeatureInstance																										
stil_block1.conceptual.sut.egi_data_sink	FlowSpecInstance																										
stil_block1.conceptual.sut.other_components	ComponentInstance																										
stil_block1.conceptual.sut.other_components.egi_data	FeatureInstance																										
stil_block1.conceptual.sut.other_components.egi_data_sink	FlowSpecInstance																										
stil_block1.conceptual.tcc	ComponentInstance																										

Figure 9 Conceptual to Runtime Traceability Report



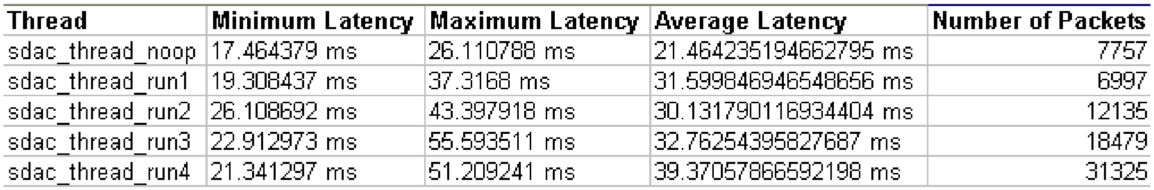
Element	Property	Property Value	Derivation Method
analysis_analysis_model_impl_instance.stil_block1.runtime.hw.eth1	AADL_Properties::Transmission_Time	47 Us .. 47 Us, 250 Ns .. 250 Ns	Estimated using times found in, ATM and Fast Ethernet Network Interfaces for Low-level Communication
analysis_analysis_model_impl_instance.stil_block1.runtime.hw.eth1	SEI::BandWidthCapacity	100.0 Mbps	Theoretical maximum of 100 Mbit Ethernet
analysis_analysis_model_impl_instance.stil_block1.runtime.hw.eth2	AADL_Properties::Transmission_Time	47 Us .. 47 Us, 250 Ns .. 250 Ns	Estimated using times found in, ATM and Fast Ethernet Network Interfaces for Low-level Communication
analysis_analysis_model_impl_instance.stil_block1.runtime.hw.eth2	SEI::BandWidthCapacity	100.0 Mbps	Theoretical maximum of 100 Mbit Ethernet
analysis_analysis_model_impl_instance.stil_block1.runtime.hw.eth3	AADL_Properties::Transmission_Time	47 Us .. 47 Us, 250 Ns .. 250 Ns	Estimated using times found in, ATM and Fast Ethernet Network Interfaces for Low-level Communication
analysis_analysis_model_impl_instance.stil_block1.runtime.hw.eth3	SEI::BandWidthCapacity	100.0 Mbps	Theoretical maximum of 100 Mbit Ethernet
analysis_analysis_model_impl_instance.stil_block1.runtime.hw.rm1	AADL_Properties::Transmission_Time	450 Ns .. 789 Ns, 22179 ps .. 22179 ps	Calculated from GE 5565 RM Node Card documentation. Assume single longword accesses.
analysis_analysis_model_impl_instance.stil_block1.runtime.hw.rm1	SEI::BandWidthCapacity	41.6 Mbps	Calculated from GE 5565 RM Node Card documentation. Assume single longword accesses.
analysis_analysis_model_impl_instance.stil_block1.runtime.hw.rm2	SEI::BandWidthCapacity	41.6 Mbps	Calculated from GE 5565 RM Node Card documentation. Assume single longword accesses.
analysis_analysis_model_impl_instance.stil_block1.runtime.hw.rm3	AADL_Properties::Transmission_Time	450 Ns .. 789 Ns, 22179 ps .. 22179 ps	Calculated from GE 5565 RM Node Card documentation. Assume single longword accesses.

Figure 10 Derivation Method Report



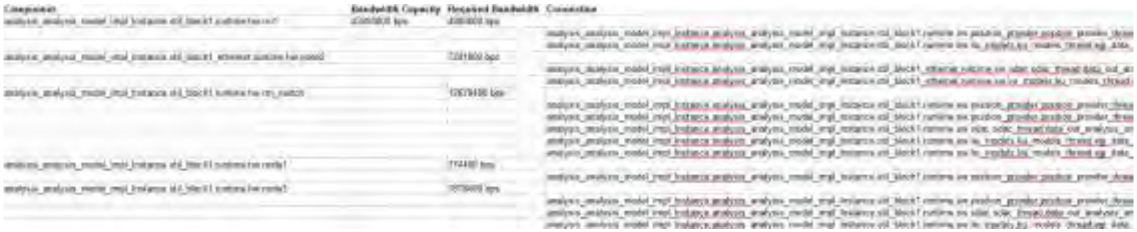
End to End Flow	Best Case Latency	Worst Case Latency	Jitter
stil_block1_ethernet.runtime.sw.pp_egi_lru_model_sdac_sink_flow	15.274 ms	53.274 ms	38.0 ms

Figure 11 Latency Analysis



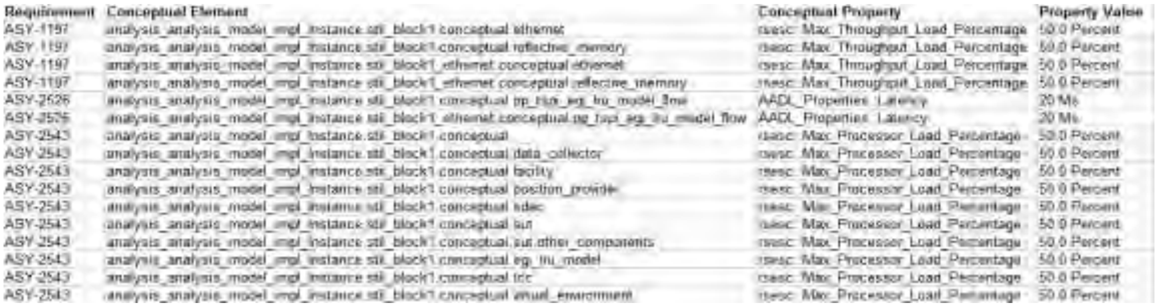
Thread	Minimum Latency	Maximum Latency	Average Latency	Number of Packets
sdac_thread_noop	17.464379 ms	26.110788 ms	21.464235194662795 ms	7757
sdac_thread_run1	19.308437 ms	37.3168 ms	31.599846946548656 ms	6997
sdac_thread_run2	26.108692 ms	43.397918 ms	30.131790116934404 ms	12135
sdac_thread_run3	22.912973 ms	55.593511 ms	32.76254395827687 ms	18479
sdac_thread_run4	21.341297 ms	51.209241 ms	39.37057866592198 ms	31325

Figure 12 Latency Measurement Report



Component	Bandwidth Capacity	Required Bandwidth	Connection
[illegible]	[illegible] bps	[illegible] bps	[illegible]
[illegible]		7201800 bps	[illegible]
[illegible]		12678400 bps	[illegible]
[illegible]		[illegible] bps	[illegible]
[illegible]		[illegible] bps	[illegible]

Figure 13 Bandwidth Analysis Report



Requirement	Conceptual Element	Conceptual Property	Property Value
ASY-1197	analysis_analysis_model_impl_instance.stl_block1.conceptual.ethernet	rsesc::Max_Throughput_Load_Percentage	50.0 Percent
ASY-1197	analysis_analysis_model_impl_instance.stl_block1.conceptual.reflective_memory	rsesc::Max_Throughput_Load_Percentage	50.0 Percent
ASY-1197	analysis_analysis_model_impl_instance.stl_block1_ethernet.conceptual.ethernet	rsesc::Max_Throughput_Load_Percentage	50.0 Percent
ASY-1197	analysis_analysis_model_impl_instance.stl_block1_ethernet.conceptual.reflective_memory	rsesc::Max_Throughput_Load_Percentage	50.0 Percent
ASY-2526	analysis_analysis_model_impl_instance.stl_block1.conceptual.pp_tspi_eg_tru_model_flow	AADL_Properties::Latency	20 Ms
ASY-2526	analysis_analysis_model_impl_instance.stl_block1_ethernet.conceptual.pp_tspi_eg_tru_model_flow	AADL_Properties::Latency	20 Ms
ASY-2543	analysis_analysis_model_impl_instance.stl_block1.conceptual	rsesc::Max_Processor_Load_Percentage	50.0 Percent
ASY-2543	analysis_analysis_model_impl_instance.stl_block1.conceptual.data_collector	rsesc::Max_Processor_Load_Percentage	50.0 Percent
ASY-2543	analysis_analysis_model_impl_instance.stl_block1.conceptual.facility	rsesc::Max_Processor_Load_Percentage	50.0 Percent
ASY-2543	analysis_analysis_model_impl_instance.stl_block1.conceptual.position_provider	rsesc::Max_Processor_Load_Percentage	50.0 Percent
ASY-2543	analysis_analysis_model_impl_instance.stl_block1.conceptual.sdac	rsesc::Max_Processor_Load_Percentage	50.0 Percent
ASY-2543	analysis_analysis_model_impl_instance.stl_block1.conceptual.sut	rsesc::Max_Processor_Load_Percentage	50.0 Percent
ASY-2543	analysis_analysis_model_impl_instance.stl_block1.conceptual.sut.other_components	rsesc::Max_Processor_Load_Percentage	50.0 Percent
ASY-2543	analysis_analysis_model_impl_instance.stl_block1.conceptual.eg_tru_model	rsesc::Max_Processor_Load_Percentage	50.0 Percent
ASY-2543	analysis_analysis_model_impl_instance.stl_block1.conceptual.tcc	rsesc::Max_Processor_Load_Percentage	50.0 Percent
ASY-2543	analysis_analysis_model_impl_instance.stl_block1.conceptual.virtual_environment	rsesc::Max_Processor_Load_Percentage	50.0 Percent

Figure 14 Requirement Traceability Report

APPENDIX E FINAL OUT BRIEF ATTENDEES LIST AND AGENDA

	Name	Organization	Phone	Email
1	Phil Zimmerman	OSD	703-681-6544	philomena.zimmerman@osd.mil
2	Crash (Kenneth) Konwin	BAH-OSD/Support Contractor	703-681-6576	kenneth.konwin.ctr@osd.mil
3	Bruce Lewis	US Army AMRDEC/SED		
4	Darrell Wright	PEO-STRI/PM-ITTS	407-384-3569	darrell.wright@us.army.mil
5	Jim Heinrich	PEO-STRI/PM-ITTS/Support Cont.	407-353-5707	jim.heinrich@us.army.mil
6	Ken Porter	PEO-STRI/PM-ITTS/Support Cont.	407-694-2479	ken.porter@us.army.mil
7	Greg Tackett	US Army RDECOM, AMRDEC/SSDD	256-876-4271	gregory.tackett@us.army.mil
8	Mikel Petty	UAHuntsville/CMSA	256-824-4368	pettym@uah.edu
9	Sue O'Brien	UAHuntsville/RSESC	256-824-6133	obriens@uah.edu
10	Philip Alldredge	UAHuntsville/RSESC	256-824-3231	pwa0001@uah.edu
11	Julie Fortune	UAHuntsville/RSESC	256-824-6314	julie.fortune@uah.edu
12	Jeff Kulick	UAHuntsville/ECE	256-824-6049	kulick@ece.uah.edu
13	Russell Peak	Georgia Tech	404-894-7572	russell.peak@gatech.edu
14	Seluk Cimtalay	Georgia Tech	404-894-8167	cimtalay@gatech.edu
15	Andy Scott	Georgia Tech	678-910-3521	andy.scott@gatech.edu
16	Barry Bullard	GTRI Huntsville	256-716-2150	barry.bullard@gtri.gatech.edu
17	Dirk Zwemer	InterCAX	404-592-6897	dirk.zwemer@intercax.com
18	Ralph Gibson			
19	Simone Youngblood			
20	Marcy Stutsman			
21	David Houseman			
22	Ed Curle			
23	Ed Weinburg			
24	Alex Boydston	US Army AED	256-313-5226	alex.boydston@us.army.mil
25	Don Ward	AVSI/SAVI		
26	Dave Redman	AVSI/SAVI		
27	Bill Paisley	Navy M&S		

Thursday January 20, 2011 Agenda

ALL TIMES LISTED ARE CST

Remote capabilities (WebEx) will only be available from 3:00 CST – 6:00 CST

Time	Event	Location
11:00 – 1:00	AHS Luncheon	UAHuntsville Bevill Center
1:00 – 1:30	Travel to the STIL on Redstone Arsenal	
1:30 – 2:30	System Test and Integration Lab Tour	Redstone Test Center
2:30 – 3:00	Travel to UAHuntsville	
3:00 – 6:00	UAHuntsville AADL-based Out-Brief	UAHuntsville Shelby Center Room 160
6:30 – 8:30	Dinner	Ol' Heidelberg, German Restaurant

Friday January 21, 2011 Agenda

ALL TIMES LISTED ARE CST

Remote capabilities (WebEx) will only be available from 7:30 CST – 11:30 CST

Time	Event	Location
7:30 – 10:30	Georgia Tech SysML-based Out-Brief	UAHuntsville Shelby Center Room 160
10:30 – 11:30	Close out Discussions	UAHuntsville Shelby Center Room 160

SYSTEMS ENGINEERING
Research Center

Verification, Validation, and Accreditation Shortfalls for Modeling and Simulation

Georgia Tech Aspects: A SysML Model-Based Approach for M&S VV&A

Principal Investigator

Russell Peak, PhD
Georgia Institute of Technology
Model-Based Systems Engineering Center
www.mbse.gatech.edu

Team Members—Research Professionals

Selcuk Cimtalay, PhD; Andy Scott; Miyako Wilson

Team Members—Undergraduate Research Assistants

Brian Aikens; Drew Martin

Contract Number: H98230-08-D-0171 **DO002, TO002, RT021**

Report No. SERC 2011-TR-018

May 3, 2011

This page intentionally left blank

TABLE OF CONTENTS

FIGURES AND TABLES

All other figure- and table-type items are included as Slides from an extended version of the final out-brief presentation (in four parts totaling ~175+ slides).

1 Introduction & Context

1.1 Context

The US Dept. of Defense (DoD) Systems Engineering Directorate in the Director, Defense Research and Engineering (DDR&E) organization initiated SERC Research Topic 21 (RT21) in the 2010 3rd quarter time frame. RT21 is entitled "Verification, Validation and Accreditation (VV&A) Shortfalls for Models & Simulations (M&S)" and is aimed at the following overall needs as described by the sponsor :

> VV&A activities for models and simulations are conducted across government, academia, and industry, both within the U.S. and internationally. In the DoD, the responsibility for VV&A exists at various organizational levels. With the recent revision to DoDI 5000.61 and past DoD-funded technical studies such as the "Risk Based Assessment", the update to the VV&A Recommended Practices Guide (RPG), and the DoD VV&A Documentation Templates as defined in MIL-STD 3022, steps have been taken to support more efficient and effective VV&A implementation. While these efforts have addressed known VV&A-related gaps, implementation of VV&A practices appears to remain uneven, sporadic, or ad hoc. VV&A research needs to focus on:
>
> 1) Users in the field, e.g., Program Managers and contractors, to better understand the gaps related to VV&A from their perspective.
> 2) Technical opportunities to address VV&A of a federation comprised of models and simulations based on 3 different perspectives:
>   a) The model/simulation operating as stand-along capabilities (single instances).
>   b) The individual model/simulation when federated (linked) with other M&S capabilities.
>   c) The collective capability of M&S assets when operating as a federation.
>
>   This is analogous to the test and evaluation (T&E) challenges of doing single system evaluation at the same time assessing its role/capability as part of a system of systems – and of the system of systems as a whole.
> 3) Methods, processes and tools to address ad hoc and sporadic implementation of VV&A, for example, methods to improve VV&A to better characterize M&S risks prior to testing.
> 4) Recommendations to minimize the risk of erroneous representations due to incomplete or inadequate VV&A.
> 5) Enterprise level efforts that can improve VV&A implementation across the DoD.
> 6) Technology, method, process or tool opportunities to advance the DoD's capabilities to perform VV&A.

As part of the RT21 team, our Georgia Tech effort in this initial phase explored how to address many of these needs using a SysML model-based approach.

1.2 Objectives and Quick-Look Approach

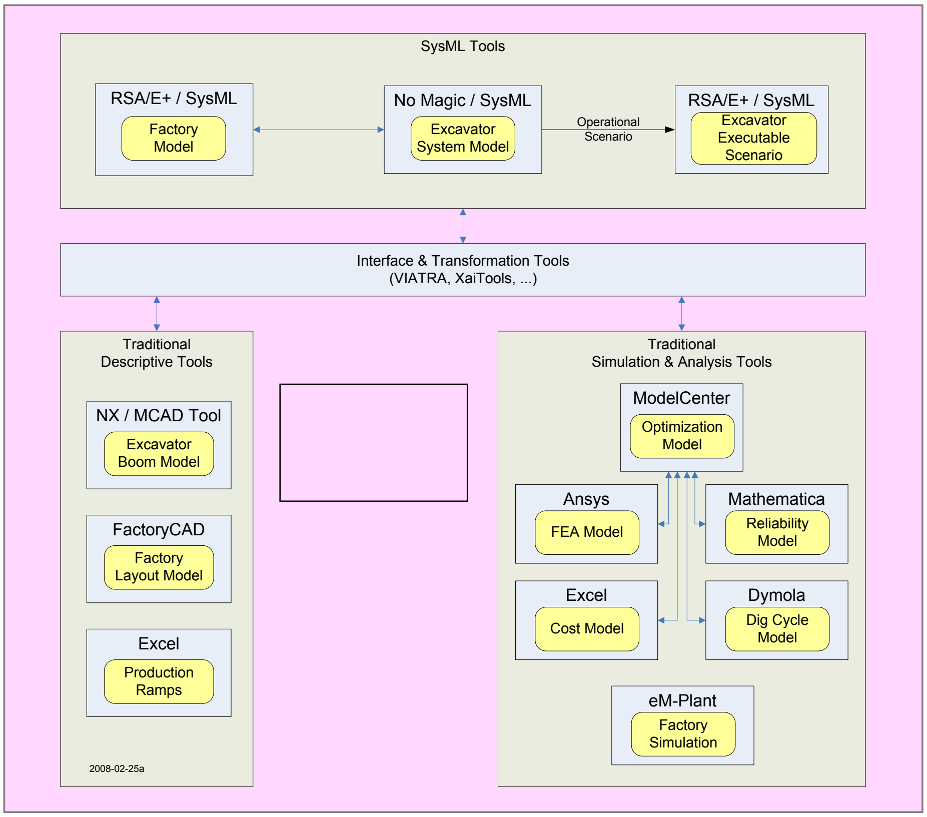
Given the needs and challenges described by the sponsor above, the Georgia Tech primary objective in RT21 has been this:

- Demonstrate how to address VV&A gaps by applying SysML and MBE/MBSE technology (especially showing how V&V can be more embedded and automated throughout system lifecycle).

Our supporting sub-objectives have been to do the following:

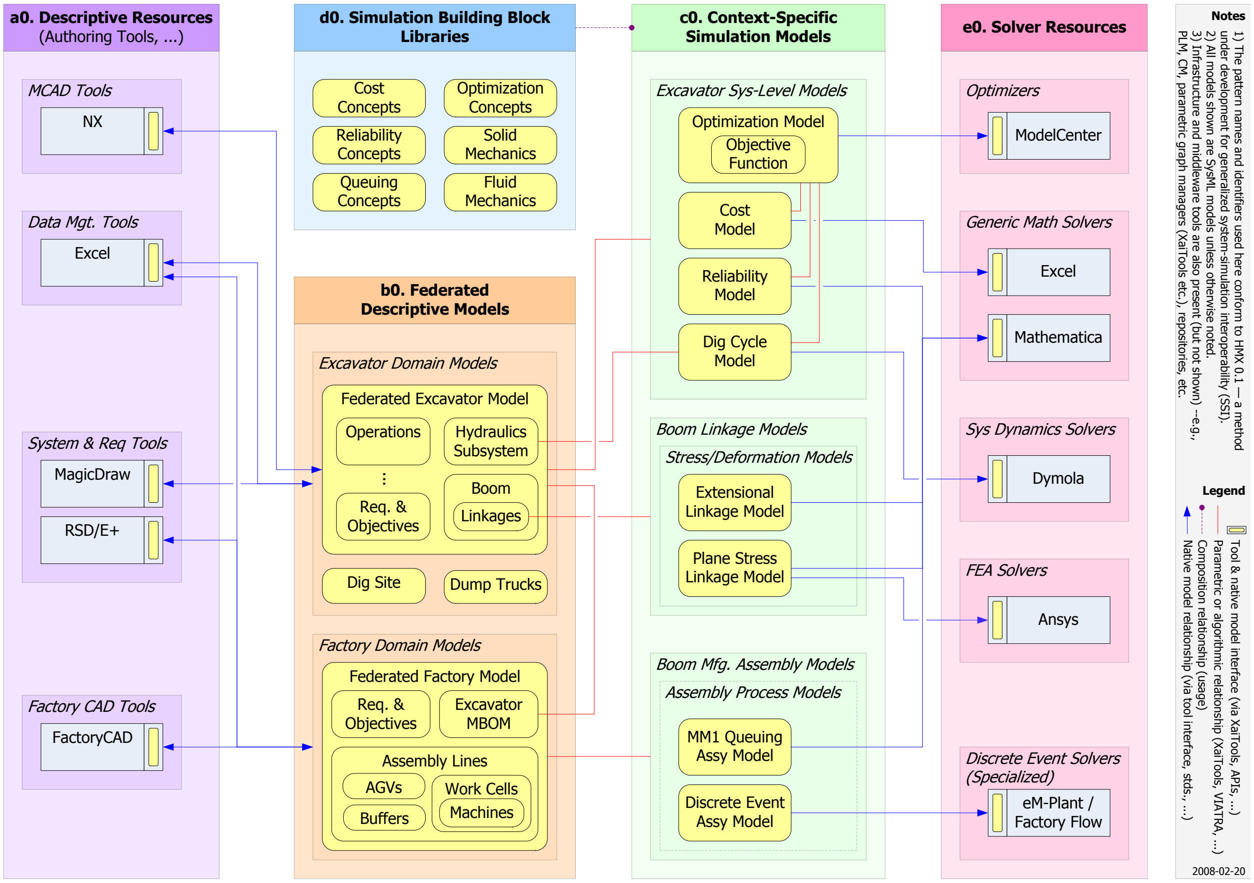
- Apply known M&S patterns and develop new patterns where needed.
- Demonstrate our approach by extending existing testbeds and examples (e.g., a excavator testbed (Figure 1 and Figure 2), a mobile robotics testbed, and other examples).
- Provide a basis (i) for developing DoD-specific testbeds and extensions in future phases and (ii) for deploying this technology for DoD programs.

Per sponsor request we have used a "quick-look" approach in this project to give the sponsor a brief broad overview how this technology can address their VV&A needs. *Therefore, we use an extended version of the final out-brief presentation as the primary content for this report and only briefly describe selected slides here.* We recommend future project phases to explore and demonstrate at least some of these topics in more depth depending on sponsor priorities.



(a) Tool categories view.

Figure 1 – Excavator testbed modeling & simulation environment. [Peak et al. 2008/2009]


a0. Descriptive Resources
(Authoring Tools, ...)
MCAD Tools
NX
Data Mgt. Tools
Excel
System & Req Tools
MagicDraw
RSD/E+
Factory CAD Tools
FactoryCAD
d0. Simulation Building Block Libraries
Cost Concepts
Optimization Concepts
Reliability Concepts
Solid Mechanics
Queuing Concepts
Fluid Mechanics
b0. Federated Descriptive Models
Excavator Domain Models
Federated Excavator Model
Operations
Hydraulics Subsystem
⋮
Boom
Req. & Objectives
Linkages
Dig Site
Dump Trucks
Factory Domain Models
Federated Factory Model
Req. & Objectives
Excavator MBOM
Assembly Lines
AGVs
Work Cells
Buffers
Machines
c0. Context-Specific Simulation Models
Excavator Sys-Level Models
Optimization Model
Objective Function
Cost Model
Reliability Model
Dig Cycle Model
Boom Linkage Models
Stress/Deformation Models
Extensional Linkage Model
Plane Stress Linkage Model
Boom Mfg. Assembly Models
Assembly Process Models
MM1 Queuing Assy Model
Discrete Event Assy Model
e0. Solver Resources
Optimizers
ModelCenter
Generic Math Solvers
Excel
Mathematica
Sys Dynamics Solvers
Dymola
FEA Solvers
Ansys
Discrete Event Solvers (Specialized)
eM-Plant / Factory Flow
Notes
1) The pattern names and identifiers used here conform to HMX 0.1 — a method under development for generalized system-simulation interoperability (SSI).
2) All models shown are SysML models unless otherwise noted.
3) Infrastructure and middleware tools are also present (but not shown) --e.g., PLM, CM, parametric graph managers (XaiTools etc.), repositories, etc.
Legend
Tool & native model interface (via XaiTools, APIs, ...)
Parametric or algorithmic relationship (XaiTools, VIATRA, ...)
Composition relationship (usage)
Native model relationship (via tool interface, stds., ...)
2008-02-20

(b) Model interoperability method (MIM) patterns view (architecture view per CT-16).

Figure 1 – Excavator testbed modeling & simulation environment. [Peak et al. 2008/2009] (cont.)

1.3 Report Structure

The extended final out-brief presentation is divided into Parts 1-4 and attached as the primary content of this report. Each section of this report describes the corresponding Part of the extended presentation (i.e., the rest of Section 1 covers the Part 1 slides, Section 2 covers the Part 2 slides, and so on).

We reference the presentation slides here in the text (instead of including them as figures) using a "Slide *p*-*n*" notation per the slide numbering in the lower right corner (where *p* is the part number and *n* is the slide number within that part). In Section 3 below (which covers presentation Part 3) we additionally use a "Slide CT-*c*-*n*" notation where *c* is the concept number (described in Section 3) and *n* is the slide number within that concept subsection.

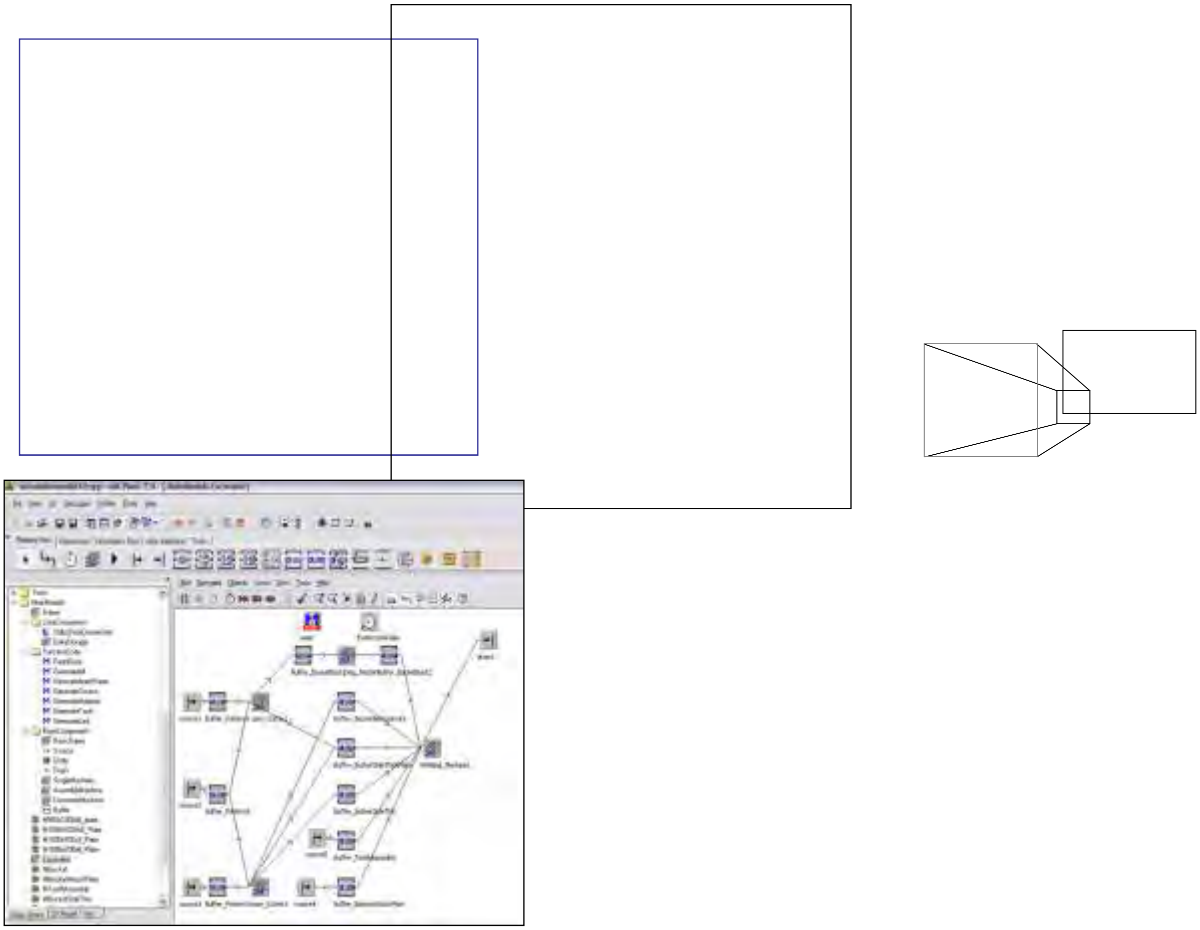


Figure 2 – Excavator testbed: sample SysML diagrams and native solver models.

1.4 Project Process and Technical Approach

We have approached the VV&A problem with a hybrid bottom-up/top-down philosophy—one that envisions building blocks for VV&A which enable VV&A activities to be ubiquitous and iterative to the degree that they become near-continuous. We believe software V&V techniques (including continuous integration[1] technologies and test-based design tools like JUnit[2]) can be generalized and applied to systems development.

Applying the Papa John's philosophy of "Better Ingredients, Better Pizza" philosophy (Slide 1-11), we hold that to get good M&S (and ultimately good systems), you must use good M&S ingredients. Hence, from a VV&A perspective, that means applying V&V from bottom to top—from the lowest-level M&S component to the highest-level simulation system—and to all

[1] See http://en.wikipedia.org/wiki/Continuous_integration and tools like Bamboo (www.atlassian.com).

[2] See www.junit.org.

M&S subsystems in between. It is with this philosophy that we have both pursued this RT21 effort and organized the this final report.

Per sponsor guidance we are leveraging existing examples from our previous work where feasible. Rather than building new DoD-based test cases in this phase, we are illustrating the technical approach in quick-look fashion with existing test cases from a variety of domains. We started by adding VV&A-oriented extensions where needed per Activity 2 in our project plan (Slide 1-12). Then in Activity 3 (Slide 1-13) we performed additional extensions and created new examples in order to fill out a sample spectrum of VV&A use cases along multiple system dimensions (including a diversity of system levels, tools, methods, and lifecycle phases). Each of the resulting concepts and examples are covered in Section 3.

1.5 SysML and MBE/MBSE Context

Slide 1-14 onward gives a quick introduction to SysML and the MBE/MBSE approach. These slides also highlight our project testbed experiences from other projects and our related short courses. Section 2 covers additional background material as prerequisite for better understanding the concepts in Section 3.

1.6 Target Impact

We conclude the [Part 1] project overview presentation with these “impact questions” from the SERC project proposal template (which is based on the Heilmeier program definition questions):

Q1: Who should care? *A1: All M&S and VV&A stakeholders (given benefits below).*

Q2: If you're successful, what difference will it make? *A2: See benefits table in Slide 1-27.*

2 SysML Concepts: Essential Prerequisites

These Part 2 slides cover essential SysML concepts including how SysML supports MBE in general and MBSE in particular (building on the introductory slides included in Part 1). These concepts, which are necessary prerequisites for understanding the rest of this report, are grouped together in these Part 2 slide set subsections:

- SysML context: system modeling & general modeling
- Representative SysML authoring tool (MagicDraw)
- Blocks and instances
- Blocks and equation-based knowledge: parametrics
  - DNA signatures (described further in Section 3 CT-11)
- Other concepts covered in later Parts as needed:
  - Requirements representation, traceability, verification, ...
  - Activities (function-based behavior)
  - Automated model-based document generation
  - Collaborative modeling environments
  - Healthy, viable, growing technology ecosystem (with many SysML users, production tools, good support, extensive learning resources, etc.)

These slides are excerpts from two courses in our short course series, SysML 101 and 102, which we have taught since Aug 2008 to over 500 participants from government and industry (see Slide 1-16).

We recommend that the reader go through the slides to gain an introductory understanding of the SysML language and its implementation in a representative tool (MagicDraw), or to refresh and see our view of the technology (especially SysML parametrics) if the reader already has some familiarity with SysML. Additional learning resources including tutorials are available at the official OMG SysML website: www.omgsysml.org.

3 SysML-based VV&A: Concepts and Examples

In this section we highlight a wide variety of SysML-based VV&A concepts. Table 1 summarizes these concepts (from CT-1 to CT-16) and lists one or more examples we utilize to demonstrate each concept. CT-17 includes several additional concepts that we believe could be demonstrated in future phases based on results to date with CT-1 to CT-16. This table shows a simplified organization scheme that starts with lower-level basic concepts and moves up to progressively higher-level concepts (which often utilize the lower-level concepts as building blocks). There are probably other good ways to break down these concepts further and to categorize them at a higher fidelity (e.g., by sub-concept, MIM type (CT-16), model type, use case type, scenario type, solution method type, and so on). Determining categorization schemes that would be most useful for various stakeholders is an area for future research.

The rest of this section describes each concept briefly and refers to its corresponding slides in presentation Part 3 where application. At the beginning of each subsection we note which MIM pattern(s) that concept primarily deals with (and we reference this further where needed in the text using a notation like this: [MIM pattern a0].

Table 1 – SysML-based VV&A concepts and examples demonstrated in RT21 Phase 1.

id	*VV&A Concept*	*Example(s)*
Core embedded V&V concepts		
CT-1	Language-level integrity: automated units consistency	MagicDraw SysML detecting units mismatch
CT-2	Language-level integrity: automated equation checking	ParaMagic detecting wrong parameter name
CT-3	Language-level integrity: other examples	Model integrity (e.g., multiplicity checking); propagating name updates; instance updates; etc.
CT-4	Augmented language-level integrity: ensuring best practices, etc.	Model checking suites in MagicDraw and ParaMagic
CT-5	Leveraging built-in checking by solvers / external tools as wrapped in a SysML context	Mathematica detecting overconstrained system of equations, etc.
Higher-level concepts – round1		
CT-6	V&V building blocks	Margin block and comparator block
CT-7	Automated requirements verification	FireSat, SimpleSat, etc. (parametrics, margin, ...)
CT-8	Embedded unit tests	LinkageSystems, build block libraries, ...
CT-9	Automated roll-up of embedded unit tests (basic multi-level test)	LinkageSystems, HomeHeatingSystem
CT-10	Automated roll-up of embedded multi-level tests	Combining above, ...
CT-11	"DNA signatures" - user interaction with model for intuitive visual inspection to aid model comprehension, V&V, debugging, etc.	LinkageSystems, FireSat/NGDMC, etc. (and above)
Higher-level concepts – round2		
CT-12	Verification of external core solvers via auto-generated native test models	
12.1	Core math solvers: Mathematica, OpenModelica, Matlab SMT	Unit test cases (to verify new solver releases, etc.);
		XaiTools production test suite (~150 models)
CT-13	Automated verification of external simulation/analysis models/tools via wrapping	
13.1	System dynamics: Matlab/Simulink	HomeHeatingSystem
13.2	Finite element analysis (FEA): Ansys	LinkageSystems
CT-14	Automated verification of external design/descriptive models/tools via wrapping	
14.1	Spreadsheets: Excel	Excavator manufacturing cost estimator
14.2	CAD: NX (MCAD); Expedition, etc., via AP210 (ECAD)	Vehicle, MiniSatellite electronics (as recorded demos)
14.3	System mission design (and LVC sims): STK	Satellite orbit & ground station comm. sys. design
CT-15	Automated verification tests on physical systems	
15.1	Activity-based test scripts with mobile robotics	Rover functionality scenarios (sensors, camera, ...)
Higher-level concepts – round3		
CT-16	MIM: an architecture for M&S patterns	
CT-17	Other concept extensions (which can be demonstrated using similar capabilities as above)	
17.1	Auto-generating documents from SysML models to support VV&A (for V&V traceability & status, accreditation reports, ...)	
17.2	Managing accreditation workflows and artifacts	
17.3	Aiding M&S validation via test results data capture and comparator usage	
17.4	Capturing SME validation criteria for future automated re-validation usage	
17.5	Managing simulation data flow and data pedigree (e.g., for sim inputs/outputs)	
17.6	Managing models & simulations themselves as systems using SysML (with requirements, structure, behavior, etc.)	
	Main Test Cases	
	- Mobile robotics (IPRE Scribbler h/w with Myro software platform)	- Excavator test bed with linkage systems
	- Satellite-to-ground station communication link simulation	- FireSat / NGDMC satellite
	- Short course tutorials (vehicle fuel system, space satellite, ...)	- Home heating system

3.1 CORE EMBEDDED V&V CONCEPTS

This subsection covers some of the most basic V&V concepts based on the "better ingredients, better pizza" philosophy described above. I.e., we show here how SysML is effective to help V&V the most basic M&S ingredients—a situation that is necessary before one can reasonably claim that higher-level M&S aspects are verified and/or validated.

A formalized language such as SysML typically includes integrity constraints such as rules that must hold for a model to be internally valid. Some constraints may be considered more at the syntactic level, which is analogous to a traditional computer language like Java having syntax rules. Other constraints are more at the semantic level, which is analogous for example to not wanting infinite loops in a Java program.

This subsection gives examples of integrity constraints at the SysML language level, at the SysML tool level, and at the core external tool level (e.g., in core external solvers that SysML uses for its parametric equations).

CT-1 LANGUAGE-LEVEL INTEGRITY: AUTOMATED UNITS CONSISTENCY

Related MIM pattern(s): cross-cutting infrastructure

Example: MagicDraw SysML detecting units mismatch

Slide CT-1-1 illustrates a key basic constraint that says units consistency needs to be maintained throughout a model. It shows two versions of a SysML parametrics diagram for part of a vehicle model. The version on the left is fine in that all variables and equations have consistent units.

The version on the right is a different story—here the modeler has accidently connected a fuel amount variable (with units of gallons) to a mileage variable (with units of miles) as seen with connection e10. And the modeler did a similar mistake with connection e11. The SysML tool automatically detects both issues and reports the specifics in the window at the bottom. It is not shown here, but the tool can also offer suggested ways to fix the problem (which may or may not be useful depending on the intent of the modeler).

This type of integrity checking may seem simplistic, but the value and impact cannot be overstated. Unfortunately significant system failures[3] and even loss of life have been caused by inconsistent units.

CT-2 LANGUAGE-LEVEL INTEGRITY: AUTOMATED EQUATION CHECKING

Related MIM pattern(s): cross-cutting infrastructure

In Slide CT-2-1 we see another automated verification capability that ensures basic language-level integrity. The slide shows the top fragment of the same SysML parametric diagram

[3] See "metric mix-up" at http://en.wikipedia.org/wiki/Mars_Climate_Orbiter, worker death during construction of the 1996 Olympic Aquatic Center, and so on.

employed as an example in CT-1. The top version is valid, but the bottom version is invalid. In this case a SysML add-on tool for parametrics solving called ParaMagic® catches the errors as seen at the bottom of the slide: the parameter names (as seen by the names near the small squares) on the eqn2 constraint property do not match the names given in the constraint expression equation. I.e., the tool automatically detects that myfuel and aa1 are not consistent with the equation definition in the constraint block. Finding such errors in a large model can be rather tedious, whereas they are automatically detected here and the SysML structure aids pinpointing them for fixing.

CT-3 LANGUAGE-LEVEL INTEGRITY: OTHER EXAMPLES

Related MIM pattern(s): cross-cutting infrastructure

This section gives several other examples showing how tools ensure language-level integrity in a SysML model. The benefit is your resulting model has a much greater chance of being problem-free at its core versus other manual approaches where it is easy for inconsistencies to slip in (e.g., PowerPoint-based modeling). A related benefit is avoiding this situation: if your model is not valid at its core, then most anything you to do with your model is suspect and may mislead decision makers.

CT-3.1 Model integrity checking: instance consistency with element definitions

Another basic language integrity feature is ensuring that an instance of an element is consistent with the structural definition of that element (analogous to a filled-out registration form being consistent with the original definition of the form, such as allowing only one person to register on a given form).

Slide CT-3-1 illustrates how a SysML tool automatically checks for such errors. The slide upper portion shows a model fragment defining that a Vehicle has two tanks called tank1 and tank2. Per the SysML spec this model indicates that these tank1 and tank2 properties can be filled in with only one tank instance each. The instance version in the lower-left is thus valid as it shows tank1=ft270 and tank2=ft280. The instance version in the lower-right, however, is not valid as it shows tank1=ft270, ft280 (i.e., two instances being used for tank1). The tool warns the modeler of this error by flagging it red and giving the message "Too many slot values" (where "slot" is a SysML term analogous to a specific blank on a specific form). This error is similar to trying to enter in two names on a conference registration form instead of using one form for each person. The tool automatically detects such issues and advises the modeler with possible options to fix them.

CT-3.2 Propagating model updates

Slide CT-3-2 illustrates another basic language integrity feature that good SysML tools support: keeping all model elements and diagrams consistent with the underlying model structure even as the modeler changes the model.

On the left is the original model that includes a Vehicle property named range. The modeler later changes this property name to distance_range (as seen in the middle of this slide). The tool immediately and automatically updates all related model elements and all diagrams that depict that property, as seen in two diagram examples in the right. Good tools will also do the same type of updates if the modeler performs more significant structural changes such as adding a new property, deleting an existing property, and so on.

This capability may seem trivial and what you would naturally expect from a tool. And you would be correct to expect that, but the most popular systems engineering tools (PowerPoint and Visio) do not even come close, and other tools often fall short (thus requiring manual update effort that is tedious, costly, and error-prone). In good SysML tools, however, automatically maintaining consistency and propagating model updates like this has become a commodity feature.

CT-4 Augmented language-level integrity: ensuring best practices, etc.

Related MIM pattern(s): cross-cutting infrastructure

While the SysML language spec has numerous integrity rules like those illustrated above, it does not necessarily cover everything a given organization may want to see. Here are two representative situations (where additional automated checking is needed to detect such problems): (a) a modeling practice is not supported by the tools a particular organization uses (even though that practice is allowed by the SysML spec), and (b) an organization has their own styles and/or domain-specific rules that they want all their models to conform to.

Here is an example of situation (a): SysML technically allows block names and property names to have spaces and other non-alphanumeric characters. But many simulation tools and math tools do not like variable names to have such characters. Thus, to ensure better robustness, reusability, and compatibility with external tools, some SysML tools provide checks for best practices like this. Slide CT-4-1 demonstrates this starting with the same valid SysML model fragment described in CT-2. The bottom of the slide shows how tools like ParaMagic automatically detect non-recommended characters and warns the modeler. Otherwise, additional errors may occur in math solvers and other downstream simulation tools that use this SysML model, and these errors typically have more serious effects and/or are more difficult to detect. The SysML spec supports extensibility and thus readily enables organizations to add tightened restrictions like this.

CT-5 Leveraging built-in checking by external solvers / tools as wrapped in a SysML context

Related MIM pattern(s): eo

In spite of capabilities like that described in CT-4, there may still be problematic situations that a SysML tool itself does not detect, but which it can facilitate detecting. One example is a SysML model that has an overconstrained system of equations. Math solvers such as Mathematica, Matlab SMT, and OpenModelica readily catch such issues. When a SysML plugin like

ParaMagic employs these math tools in an automated manner, it can ask the math tool to tell it if anything went wrong during the solution process. Thus the SysML-based approach leverages these external capabilities in a structured context, and it does not need to re-invent the wealth of knowledge and V&V that such solvers already have.

3.2 HIGHER-LEVEL CONCEPTS – ROUND1

CT-6 VERIFICATION & VALIDATION BUILDING BLOCKS

Related MIM pattern(s): e0

This section shows how you can create V&V building blocks that are quite useful and applicable in numerous contexts. These blocks capture several basic V&V concepts in a modular reusable form. They are fundamental elements for V&V just as basic electrical elements (such as resistors, capacitors, transistors, and diodes) are fundamental to electrical circuits. We highlight two such blocks here: a margin block and a comparator block. We also briefly note other similar blocks and variations that could be similarly created to provide practitioners a more complete set of V&V building blocks.

CT-6.1 Margin block

Slide CT-6-1 shows the SysML definition structure for a margin block. The SysML parametrics diagram (par) on the left gives the primary margin equation (r1) and also a verdict test equation (r2). As explained in the note on this diagram, margin quantifies the comparison between an maximum allowable value versus some determined value. One common application of this concept is the case where the maximum allowable value is dictated by a requirement (e.g., maximum allowable mass) and the determined value is calculated based an analysis model or simulation model for the current design. In other cases the determined value is found by some other means such as a physical measurement as part of a test on the current system.

The margin concept is commonly employed in the aerospace industry to help quantify design status with respect to requirements and objectives (e.g., in stress analysis to see how much margin a wing structure has before exceeding its material yield stress). We illustrate this type of application in the CT-7.1 section below to automatically verify if a satellite system design meets several requirements dealing with mass and power limits.

Other engineering disciplines similarly utilize margin and/or related concepts such as factor of safety. Note that these concepts are also useful for non-physical systems, for example, to compare the actual determined cash-on-hand that a business has versus its desired level of cash-on-hand that gives a buffer for cash flow purposes.

The margin block implementation shown here handles perhaps the most common needs, but there are other useful variations it does not support including minimum allowables, ranges, tolerances, multiple levels of acceptability (not just pass/fail), expected delta trend versus baseline, and so on. Thus, one recommendation is that the sponsor consider supporting the development of V&V building block libraries that would include a whole suite of such blocks (as well as similar variations for the comparator concept described next). Making such libraries readily available to the DoD supply chain is one way to foster widespread practice of this approach and to achieve the related benefits: *Build a useful library and they will come!*

CT-6.2 Comparator block

The comparator block illustrated in Slide CT-6-2 has similar concepts and intent as the above margin block. The main difference is comparators are mostly used to check for equality, whereas margin blocks expect to have a non-zero delta between determined and allowable. A comparator automatically verifies if an actual value is equal to an expected value. Thus comparators can be employed for M&S V&V purposes practically everywhere, for example, to verify that a simulation produces expected values under specified circumstances.

We demonstrate this usage below in several examples (CT-8.1, CT-9.1, CT-13.1, etc.) to verify design/analysis/simulation models ranging from mechanical parts to heating systems to space systems. At Georgia Tech we use this approach in conjunction with InterCAX LLC to automatically verify software libraries and SysML models that are utilized in the industrial-grade products that InterCAX sells commercially (with customers including NASA JPL, Sandia, and many 1st and 2nd tier companies in the DoD supply chain).

CT-7 AUTOMATED REQUIREMENTS VERIFICATION

Related MIM pattern(s): b0, c0, d0

This section shows how SysML enables automatically verifying requirements. This approach can be applied to requirements for physical systems as well as to requirements for M&S.

CT-7.1 Using margin blocks—SimpleSat example

These slides show the SysML-based design of a basic satellite, which we use for teaching SysML concepts. It employs the margin block introduced above (CT-6.1) to verify three requirements (as seen by the three black diamond part property relations connected to Margin Block in Slide CT-7-1). Slide CT-7-4 shows a solved and verified design state as displayed in the ParaMagic browser. The resulting margin value (denoted mos in this model version) in each margin block indicates that this design meets the mass requirement (by 1% as seen in reqVerifierMass) and the power subsystem power requirement (by 11% as seen in reqVerifierPower), but fails the controller subsystem rating requirement (by a negative 9% as seen in reqVerifierRating).

CT-7.2 Using direct pass/fail expressions—FireSat/NGDMC example

This example (Slide CT-7-5) is based on a SysML implementation of the FireSat satellite design described in the widely-used *Space Mission Analysis and Design* (SMAD) handbook by Wertz and Larson plus extensions for NGDMC. Slide CT-7-6 shows a solved and verified design state as displayed in the ParaMagic browser. In this model the system engineer uses direct conditional expressions (instead of margin blocks) to automatically verify the status of two requirements: one for cost (cosRt1) and another for satellite scan area coverage (covRt1). A value of one (1) for each corresponding verdict attribute (CostReqtVerify and CoverReqtVerify) indicates the system design passes these requirements (versus a zero (0) would indicate it fails).

This direct conditional expression approach is also effective modeling style, though in general an explicit margin block approach is more recommended because it has better modularity and reusability.

CT-8 EMBEDDED UNIT TESTS

Related MIM pattern(s): do

This concept applies software unit test thinking (as see in technologies like JUnit for Java) to the design of generalized systems, where a system can be an M&S system (or practically any type of system model) and its building blocks. The main idea explored here is adding automated unit testing at every system level—from the lowest-level building block to the highest-level system-of-systems—thus better ensuring verification at each level (as well as some types of validation as discussed in CT-17).

CT-8.1a Using comparator blocks—linkage system example

Slide CT-8-1 illustrates the basic SysML-based unit test pattern, which has the element-under-test (EUT) (a.k.a., the system being verified) and (UT) the unit test itself, which uses the comparator block structure described in Section CT-6.2 one or more times like virtual test probes (TP_j). In this case there are seven comparator block usages (as denoted by cmp01 to cmp07), each of which compares an expected value (from a golden instance) with the current actual value of a key system property.

You could conceivably apply an exhaustive form of this pattern and use a comparator block for each and every property in the EUT. The exhaustive comparison approach is recommended at the lower building block levels, but that could get tedious, redundant, and/or not impractical for large systems. So the approach used in this example is like attaching test probes to seven key properties in the system. Since many of the properties are interdependent in some way (seen DNA signature in Slide CT-8-3), you are likely to catch any issues in the system by checking the values of these seven test probes. Future project phases could investigate ways to auto-generate the right test probes for a desired level of risk (e.g., based on sensitivity analysis or related testing algorithms from the electronics and software domains).

CT-8.1b Using comparator blocks—home heating system simulation example (Simulink)

See also Slide CT-13-3 (and related slides) to see how we applied this same unit test pattern to a home heating system simulation based on a Simulink model.

CT-8.2 Using comparator blocks—building block examples

There are no slides for these examples, but you can probably readily visualize how the same unit test concept applied to the above linkage system could be applied to practically any model level. Thus, you can apply unit testing to the libraries that contain the lowest-level building blocks, including the sleeve block used in the linkage system, as well as the circle block used in the sleeve block, and so on down to the most primitive building block. A key point is that you can do this testing at the library level independent of any particular system built from that library. E.g., the golden instances you use to test building blocks like sleeve and circle would normally not be the sleeve and circle instances that any particular linkage system instance utilizes.

We recommend to embed unit tests like this starting with the most primitive building blocks and working up from there. This way you can verify all your internally- and externally-supplied libraries (applying CT-9) before using them in your system modeling (and you can automatically re-verify them whenever something changes, such as when new editions of your 3rd party libraries become available).

CT-9 AUTOMATED ROLL-UP OF EMBEDDED UNIT TESTS (BASIC MULTI-LEVEL TEST)

Related MIM pattern(s): do

This concept connects together two or more unit tests like those described in the previous section (CT-8). Slide CT-9-1 shows this pattern (similar to the above unit test (UT) pattern) applied again to linkage systems, but now a multi-unit test (MUT) block takes advantage of the unit test blocks already defined. In this linkage system case, this multi-unit test is verifying two linkage system instances and rolling up the combined result. The MUT verdict is thus pass only if both unit tests are pass.

In this way you can define a suite of unit tests for better test coverage (e.g., a UT for one instance that has values at one extreme, another UT for an instance with values at the other extreme, and several UTs for other instances with special case values, etc.). We have similarly demonstrated this pattern with the home heating system (not shown here) applying the UT seen in CT-8.1b.

CT-10 AUTOMATED ROLL-UP OF EMBEDDED MULTI-LEVEL TESTS

Related MIM pattern(s): do

By extending the multi-unit test concept a bit further, you can probably easily imagine rolling up automated testing like this to practically any level such that tests in a parent system would automatically run all tests in all its child subsystems and so on down the line. One strategy is to have (a) a "subset roll-up test network" of embedded tests like this as a quick check (where this roll-up test network includes just one or two tests at each level), and (b) an "exhaustive roll-up test network" that runs all tests at all levels. Another strategy is to leverage the same kind of thinking that is found in continuous integration systems for software development (where automated tests like these are continuously run whenever triggered by a system change).

CT-11 "DNA SIGNATURES" - USER INTERACTION WITH MODELS FOR INTUITIVE VISUAL INSPECTION TO AID MODEL COMPREHENSION, V&V, DEBUGGING, ETC.

Related MIM pattern(s): cross-cutting infrastructure

First we provide a brief introduction to the DNA signature[4] concept referenced in Section 2 in relation to SysML parametrics. We assume the reader has already become familiar with the basics of SysML parametrics by reviewing Part 2 slides and the V&V concepts thus far (or via equivalent background material). Slide 2-19 (in the Part 2 presentation) shows the primary DNA signature nomenclature and how it maps to SysML parametrics nomenclature. Briefly, a yellow symbol in a DNA signature represents a SysML value property, a blue symbol represents a SysML constraint parameter, and a red symbol represents a SysML constraint property. An informal description for each of these is shown in parenthesis: a system attribute, a local variable, and an equation usage. The fuel tank example shown in Slide 2-19 has only three value properties and one constraint property, and thus its DNA signature closely matches its SysML parametric diagram.

The building block nature of SysML and the usefulness of the DNA signature view becomes more evident in the vehicle example in Slide 2-25. The Vehicle block uses the Fuel_Tank block twice as two part properties named tank1 and tank2. And the Vehicle block has five constraint properties (representing five equation usages) that together determine several vehicle-level fuel and range attributes. Note however, that the vehicle DNA signature (Slide 2-26) has seven red symbols (indicating seven constraint properties) instead of five—why is this? It comes from the vehicle having two fuel tanks:

(5 eqns. in Vehicle *block) + (1 eqn. in* Fuel_Tank *block) x 2 part properties = 7 eqns. total*

Keep in mind that this "block built from blocks" pattern can be nested arbitrarily deep (while in this case there is only one level of nesting). In a nutshell that is how you can get complex DNA signatures fairly quickly even though the SysML parametric diagram for any given block may not look very complex (see examples in the slides starting with Slide 2-28 in Part 2). In fact it is good practice for any given block to define no more than 5-10 equations at its own level—beyond that it should start using build blocks to capture additional equations and so on.

Model builders find that SysML blocks and their parametrics aspects are helpful for constructing a model, and that DNA signatures (which are auto-generated from their resulting model) are helpful for debugging and understanding the total effect of the emergent model. Thus, using these various views together facilitates both composing and comprehending complex models. Next we describe how DNA signatures aid doing V&V for M&S in particular.

CT-11.1 Verification of building blocks and higher-level models

When you create modular building blocks using a technology like SysML, it helps to have a quick visual means to verify them. The most primitive (leaf-level) building blocks tend to have ten (10) or less equations, and thus their DNA signatures are relatively simple and quickly recognizable. Creators of building block libraries (and also users of those libraries) come to learn their building block DNA signatures and immediately recognize at a visual glance when something has changed. Hence, DNA signatures provide an intuitive visual means for

[4] DNA signatures are not officially part of SysML 1.x (though it technically could support a similar view, but in practice no tools support that yet). We developed this view in our research based on our experiences with SysML parametrics (and its foundational composable object concepts) to provide benefits like those noted in this section. If more and more people become familiar with DNA signatures and find them useful like we do, perhaps they will become an official part of a future SysML spec release.

verification by humans. They give new capabilities and depth to enable the saying "something does not look right".

The same can be said for higher level complex models built from these primitives. If model creators and users spend some time with a specific complex model, they get to know what it looks like (and what its subsystems and primitives look like) via its DNA signature. At higher levels it can be admittedly difficult to detect subtle changes, but they can readily see changes that have greater impact (and they can always isolate and view any single subsystem or building block to verify it by itself).

CT-11.2 Validation via expected/unexpected structure

One way DNA signatures help subject matter experts (SMEs) validate a model is by enabling them to see expected and/or unexpected structure (Slide CT-11-1). For example, when someone was recently building a SysML model for sizing satellite systems, the DNA signature showed two distinct (disconnected) subgraphs: one showing the equation relations among mass-type properties and another showing the same for power-type properties. A satellite SME could see right away (without inspecting any further details whatsoever) that the model was not a good model of satellite system reality—the DNA signature clearly indicated that the model did not consider mass and power to have any effect on each other (whereas in reality the power system solar panels and batteries do indeed have mass). An SME would expect the model to relate power and mass somehow, which would readily show up in the DNA signature as one or more visual connections in the equation graph. The DNA signatures in CT-9.1 and CT-8.1a provide additional examples, as they have several disconnected subgraphs and even some dangling value properties (not connected to anything)—these situations may raise flags that warrant further investigation (depending on the intent of the model and its domain semantics).

Similarly there can be cases where the DNA signature contains unexpected connections that would cause an SME to investigate further and perhaps detect an invalid aspect of the model. Continuing the satellite example, if the SME sees a direct connection between battery mass and solar panel power generation, there is probably something wrong (since the former has more to do with energy storage and not energy generation, and thus there should be one or more mediating relations in between). In summary DNA signatures give SMEs a transparent way to inspect and trace a model from its highest level structure to its lowest level building block (and to all points in between).

CT-11.3 Validation via common generic patterns

Another way DNA signatures help SMEs validate a model is by enabling them to look for expected patterns. This is similar to CT-11.2 except now they are looking for one or more overall "gestalt" patterns that are common to many types of model (versus the more model-specific connections in CT-11.2).

For example the "pinwheel" is a common pattern you can expect to see whenever you have a roll-up type of calculation at any given system level. Slide CT-11-1 shows this pattern for a recycling facility, where each "arm" of the pinwheel is one recycling process that an electronics device goes through to get disassembled and recycled. The center of the pinwheel is the energy roll-up equation (which indicates how much total energy the facility uses across all its processes).

Thus, SMEs can look for this pinwheel roll-up pattern in a model in their domain as a quick way to check if the model has needed capabilities with respect to whatever roll-ups are important for that domain (e.g., rolling up things like cost, mass, time, capacity, and so on). As people become more and more familiar with DNA signatures and similar concepts, other similar gestalt patterns are likely to surface and find similar utility for V&V purposes.

3.3 HIGHER-LEVEL CONCEPTS – ROUND2

Next we look at several more higher-level concepts that build upon what we have seen so far.

CT-12 VERIFICATION OF EXTERNAL CORE SOLVERS VIA AUTO-GENERATED NATIVE TEST MODELS

Related MIM pattern(s): e0

CT-12.1 Core math solvers: Mathematica, OpenModelica, MathWorks Matlab SMT

As seen in Section 2, SysML models can use tools like ParaMagic drive the auto-generation of complete analysis and simulation models. In this concept section we highlight how to leverage that capability to automatically verify the solvers themselves (and related variations upon that theme).

Use Case 1: Verifying different solvers. This case assumes that you use one or more solvers S1, S2, ..., Sn as part of your M&S environment (e.g., where S1= Wolfram Mathematica, S2=OpenModelica, and S3=MathWorks Matlab Symbolic Math Toolbox (SMT) in our testbed environment). You can define an equation-based model, EM1, as a SysML model that should be solvable by S1..Sn. You can also determine known-good output results for EM1 for a specific set of inputs (e.g., by doing manual calculations, by finding known-good results from other independent sources, or by some other means).

You can then use an orchestration tool like InterCAX ParaMagic to solve EM1 using solver S1 and confirm that S1 produces the expected results. You can also use the exact same SysML model EM1 to do the same verification for solvers S2, S3, ..., Sn. By doing this for enough models EM1..EMp that cover your M&S class(es) of problems, you have effectively verified your solvers to handle those class(es) of problems. The value of doing this with a SysML-based approach becomes more evident as you add more solvers Sj and more SysML-based models EMi.

Slides CT-12-1 onward illustrate this approach for a classical basic test model (an analytical spring system) [Peak et al. 2007] and the corresponding auto-generated input job files (and output file results) for the solvers S1, S2, and S3 listed above. We use this iterative approach at Georgia Tech in collaboration with InterCAX to verify these solvers using ~150 different models (EM1..EM150). These models cover numerous SysML modeling features, mathematical situations, and complexity ranges (from quick tests that run in seconds to scalability tests that run in 100's of minutes; from models built internally purely for test purposes to production models built by external organizations).

Use Case 2: Verifying deviations from a baseline (deviations in solvers, SysML authoring tools, models, orchestration tools, etc.). Once you have a baseline configuration established and verified per Use Case 1, you can then use it to verify several types of variations. For example, when a vendor releases a new solver version, you want to make sure it still works fine for your types of problems. Rather than expending costly error-prone manual effort, you can auto-re-run your SysML model suite to ensure you get the same expected results. Of the ~150 test cases above, it is not uncommon to find 1 or 2 test cases that have issues with the new solver version (which then typically results in bug reports and the vendor needing to release a patch to fix their solver, or sometimes an update to our tools due to changes in vendor formats).

Similarly, whenever we update our models SysML-based EMj themselves, we re-run the affected suite portions to make sure these models themselves still work as expected. And we do the same for new versions of our *XaiTools FrameWork* (which is the basis for the commercial InterCAX orchestration tools including ParaMagic, ParaSolver, Melody, and Solvea) as well as the orchestration tools themselves.

CT-13 Automated verification of external simulation/analysis models/tools via wrapping

Related MIM pattern(s): e0

This concept is similar CT-12 except that in this case the external simulation/analysis model is a pre-existing model that SysML is wrapping in a black box manner. I.e., in CT-12 the solver models were fully and automatically generated from a SysML model, but in this CT-13 case, the we simply wrap an existing external model EMj—providing it inputs and then checking that the outputs match expected values (without the SysML model needing to know much if anything about the internals for EMj). Note that some wrappings can be a bit more gray than black if some degree of model structure is replicated (or mapped to) in the SysML model.

This concept supports several helpful use cases to V&V that an external simulation/analysis model meets expectations, including (a) comparing the model against known-good results as determined from some other source, and (b) comparing the model against previous versions of the same model and/or tool (e.g., ensuring that the model results for Matlab 2010a are consistent with what you had before in Matlab 2008b). Applications with a few representative simulation and analysis tools are described next.

CT-13.1 System dynamics: MathWorks Matlab/Simulink

Example: HomeHeatingSystem. These slides highlight how CT-13 works with a Simulink model (a time-based thermal system dynamics model in this case, which also includes operating costs). Note we apply here a similar comparator block-based unit test pattern as seen in CT-8.

CT-13.2 Finite element analysis (FEA): Ansys

Example: LinkageSystems. Slide CT-13-5 highlights how CT-13 concepts similarly work well for an Ansys model (a 2D stress and deformation analysis model in this case). The native Ansys model is based on a parameterized script, and the SysML wrapper provides input/output access to a selected subset of those parameters. The FEA model creator may have many parameters designed into the FEA model, but only a dozen or so of those in this case are of interest in systems engineering and V&V contexts. Note the usage here, too, of the CT-7 pattern (using margin blocks), and also how we could apply CT-8 to wrap this in a unit test to verify Ansys itself.

CT-14 Automated verification of external design/descriptive models/tools via wrapping

Related MIM pattern(s): a0

This concept is quite similar CT-13 except that in this case the external models/tools are intended primarily for design purposes [MIM pattern a0] (whereas in CT-13 the external models/tools are primarily for analysis or simulation purposes [MIM pattern e0]). Some cases could be argued to be a combination of both design and analysis/simulation (as some tools mix the two functionalities). Like CT-13, these employ SysML-based wrappings around existing models in a black/gray box manner (where the wrappings connect only to key inputs and outputs, thus leaving most of the external model details to be handled by the native tool). Applications with a several representative tool varieties are given here.

CT-14.1 Spreadsheets: Microsoft Excel

Example: Excavator manufacturing ROM cost model [Peak et al. 2009]. Similar to the CT-13 examples, this example (Slide CT-14-1) illustrates wrapping a spreadsheet as a type of external model where the SysML context provides inputs and reads outputs. Thus, we can apply the same unit and multi-level comparator block test probe patterns seen in CT-8/9/10 for V&V purposes.

CT-14.2 CAD: Siemens NX (MCAD); Mentor Graphics Expedition, etc., via AP210 (ECAD)

Example: vehicle geometry; MiniSatellite system & electronics (also available as recorded demos). These examples (starting with Slide CT-14-2) illustrate similar capabilities as the other CT-14 examples, except they use a somewhat different wrapper/interface approach (thus illustrating that SysML supports a variety of approaches).

The other examples have input/output interfaces that are configurable at the native model scripting level (or even the coded plugin level in the case of the current CT-14c prototype). Thus, they take a bit of work outside the SysML model for their initial setup, and then after that they are highly automated.

These examples leverage an approach that is a bit more flexible. The interface plugin establishes the SysML wrapper input/output structure automatically based on the native model and then keeps automatically synchronizes them in an on-demand manner. From that point onward he effect and usage for V&V purposes is essentially the same as the other examples. The MiniSatellite example combines both MCAD and ECAD wrappers and shows how system parameters like cost and mass can be rolled-up across multiple system/subsystem levels and used to auto-verify requirements (by also applying CT-7 concepts), thus demonstrating executable traceability from the top-level system model down to the lowest-level domain-specific model (ECAD and MCAD in this case).

CT-14.3 System mission design (and LVC sims): AGI STK

Example: Satellite orbit/trajectory & ground station system design. The purpose of this test case was determine if SysML could help V&V the more traditional types of live/virtual/constructive (LVC) simulations. AGI STK is a commercial LVC simulation toolkit and environment that is widely used in industry and government (see Slides CT-14-9 and -10) . While its origins are in satellite orbit/trajectory design and simulation, today it is employed in a wide variety of applications ranging from ISR to electronic communications.

In this project we developed a prototype plugin interface between SysML (MagicDraw) and STK. This initial plugin supports two-way interoperability during simulation run-time, where the inputs (from SysML to STK) are indicated by the blue arrows (Slide CT-14-11) and the outputs are the red arrows. The test case simulation involves an earth-orbiting satellite and two ground stations (one in South America and one in Australia as their initial locations).

STK automatically displays a graphical link and calculates communication link duration for each communications occurrence (i.e., for each period that the satellite has line-of-sight contact with one or both ground stations). The user can change ground station and orbit design parameters in the SysML model (e.g., changing orbit inclination angle, ground station position and elevation, and so on) and immediately see the effect on simulation visualization and link duration results.

The plugin reads communication link durations and updates the SysML model instance after each communication link occurrence is done. Slide CT-14-12 shows how this updated instance can then be used as part of the higher level systems context just like all the other examples in this report. In particular you can run SysML parametric calculations on the link durations for purposes including requirements verification and simulation validation. For example, an SME might do this test to see if the simulation behaves as expected: Make the satellite going in an orbit around the equator by setting inclination angle to zero. Then the link duration times should remain constant (within simulation round-off error) since the lines-of-sight should be constant. We can capture this type of SME knowledge as automated test cases and run them to help V&V the simulation.

Thus this test case demonstrates how SysML can also be effective for M&S V&V for traditional LVC-type environments (with this example being more of the constructive type). Future project phases could further test and demonstrate this capability with varieties of each LVC type using a similar approach.

CT-15 Automated Verification Tests on Physical Systems

Related MIM pattern(s): a0, b0

This section introduces how it is also possible for SysML models to aid in doing V&V on physical systems themselves (in addition to V&V'ing the design and simulation models of that physical system, as the cases above are focused on).

CT-15.1 Activity-based test scripts with mobile robotics

Example: Rover functionality scenarios (sensors, camera, ...). The slides show a small rover system that is used at several universities for computer science education purposes (see www.roboteducation.org). We adopted this rover platform ~2 years ago for SysML research and education purposes, namely by adding a plugin to MagicDraw so that you can operate a rover as prescribed by your own user-created SysML activity models.

Because we can both control the inputs to the rover and read outputs from the rover (such as battery level, light sensor levels, camera images, etc.), we can similarly use SysML activities to automate at least some types of V&V on one or more actual physical rovers. For example, we could make the rover travel 10 repetitions of a prescribed path and verify that the battery level drop is within an expected range.

We could also use this approach to help V&V M&S models about the rover. For example, we could have a model that predicts battery level drops, then we could auto-run several rovers and read in actual measurements of battery level and compare model results vs. measurements to help validate the model. Thus this example illustrates an additional capability dimension beyond using SysML to manipulate or wrap just the M&S model itself.

3.4 Higher-Level Concepts – Round3

This section covers additional concepts that build upon and extend the above concepts.

CT-16 MIM: An Architecture for M&S Patterns

This section highlights an architecture for generalized model interoperability patterns that are becoming increasingly evident based on experiences from our INCOSE MSI team testbeds and related work [Peak et al. 2009; Peak et al. 2007]. This section outlines this emerging architecture, denoted MIM (model interoperability method), which builds on both past [Peak et al. 1998; Tamburini et al. 2005] and current research. By understanding and recognizing these patterns, one can more effectively plan, specify, design, implement, verify, deploy, and maintain heterogeneous modeling environments (such as Figure 1) in a wide variety of contexts (i.e., beyond excavators), together with the corresponding model-based engineering practices.

MIM applications—diverse examples

Slide CT-16-1 lists diverse applications of the MIM approach to date, and several are illustrated in the next several slides (most using a composable object notation which pre-dated SysML and provided the primary basis for SysML parametrics). We highlight two applications here.

MIM applications—excavator systems

Figure 1(a) illustrates an excavator testbed in a recent project that demonstrates the MIM approach (Peak et al. 2009). Three distinct categories of commercial-off-the-shelf (COTS) tools are employed: SysML tools, traditional descriptive tools (product CAD, factory CAD, and Excel), and traditional analysis tools (Excel, FEA, math solvers, and simulation solvers). In addition, the project team developed the necessary interfaces to integrate the SysML modeling tools and other tools using a combination of COTS interfaces (e.g., VIATRA/MOFLON and ParaMagic®) as well as custom interfaces developed in C#, Java, and Visual Basic.

Figure 1(b), also given as Slide CT-16-2, illustrates the resulting models and model interfaces from a MIM model interoperability patterns perspective. These patterns are key enablers for platform-based engineering (PBE) and model-based engineering (MBE). On the left are COTS descriptive tools (labeled a0), and on the right are COTS solver tools (labeled e0). The models in the middle boxes (labeled b0, c0, and d0) are implemented as SysML models (Figure 2). The box labeled b0 is the federated systems model which is a descriptive-type model that collects together various a0-type models and augments them where needed. In this testbed the b0 model combines both the system-of-interest (excavator products) and its manufacturing system. Throughout the project, reusable analysis and simulation building blocks that are context-independent (generic) were identified and collected into libraries, as illustrated in the PBE-oriented box labeled d0. Each context-specific simulation model in Figure 1(b) (box labeled c0) applies selected generic d0 building blocks to the b0 system for a specific purpose—typically to calculate values to verify one or more requirements or performance objectives.

Each c0 model is executed utilizing one or more e0 solvers, which are typically general purpose COTS solvers, but may also be specialized company-proprietary codes. The c0 model pattern is the focal point for capturing knowledge about domain-specific analysis intent including idealization decisions. Depending on the nature of the b0 system aspect being analyzed, these c0 models range from fixed topology analysis templates (which analysts create directly) to variable topology analysis templates (which auto-generate a model with simulation topology that is specific to a particular design instance). In our excavator testbed the boom linkage models are examples of the former, and the dig cycle hydraulics model is the latter. Each arrow in Figure 1(b) represents a specific interface that required development, implementation, and testing by the project team. SysML modeling and interface development represented the major part of the R&D effort for the project. To demonstrate the model integration illustrated in Figure 1(b), a series of scenarios were created for the excavator example. Figure 2 contains several thumbnail highlights from these scenarios (see the project report [Peak et al. 2009] for further explanation). The initial scenario represents a design requirement (a target rate for moving dirt), and a marketing requirement (a target rate for selling excavators). The hydraulic and structural teams exercised a design process to achieve a satisfactory product design, and the manufacturing team translated the design into a manufacturing plan, capacity plan, and operational plan. The design process was then confronted with changed requirements. A higher required rate for moving dirt was found to necessitate a redesign of both the hydraulic and the structural subsystems; this revised product design then required a redesign of the manufacturing process. Finally, a higher target sales rate required a further redesign of the manufacturing process to support the corresponding increased manufacturing rate.

Per these foundational results, we demonstrated how to bridge a SysML system specification and design model with multiple engineering analysis and dynamic simulation models—i.e., the overall excavator project objectives was met and exceeded. The significance of these results is not just in any single design decision or supporting engineering analysis—all of these could be done individually without the SysML modeling and interface development, albeit via ad-hoc less effective means. Rather, the significance is in the formal capture of modeling and design knowledge in a manner that enhances both design and analysis integration and knowledge and information reuse. Integration fully or partially automates time-consuming manual processes and thus enables faster design analyses with less effort by the designer, which can result in both faster design cycles and increased design analysis and trade space exploration. Knowledge capture and reuse enhances the capability of designers in terms of both design speed and design quality. The impact of improved model management is better visibility and communication across the entire design/manufacturing cycle, leading to fewer errors, earlier problem identification, and faster problem resolution.

This knowledge capture, integration, and reuse occurs at two levels. First, at the domain level, knowledge capture takes the form of libraries of concepts, modeling elements, and interfaces that are directly reusable in the design of other excavator products or other excavator manufacturing processes (and often in other domains beyond excavators). The use of these libraries does not necessarily require expertise in SysML, i.e., the captured knowledge can be accessed by potential users in “wizard” forms or in tools with which they already are familiar. Second, the captured knowledge takes the form of explicit system models that integrate the design across multiple product design disciplines and between product design and manufacturing. The demo scenarios show how knowledge is captured and enables very rapid and inexpensive redesign of both the product and its manufacturing processes.

MIM applications—linkage system tutorial

Slides CT-16-10 onward illustrate the MIM architecture and its main patterns using a comprehensive linkage system tutorial. This tutorial is based on the seminal SysML parametrics papers by Peak et al. [2007, Parts 1 & 2] plus updates to use the newer more generalized MIM terminology (vs. the earlier MRA terminology, which was focused on patterns for design-analysis integration [Peak et al. 1998]). Bajaj et al. [2011] describe the SLIM approach, which could be viewed as a superset context for MIM.

MIM objectives

From one perspective, the main objective of MIM is that it should specify, design, implement, and verify the interfaces needed to support model interoperability throughout the system lifecycle. It may also include how the overarching MBSE method should apply the interface to achieve the purpose. This perspective provides an entry point where people may often start by answering the question "how do I connect model type A to model type B?".

For example if someone has a different SysML tool (e.g., Rhapsody) and a different modeling application to tie to their SysML model (e.g. STK), then they could look to MIM to guide them how to connect these tools and associated models. I.e., they could use MIM to specify, design, implement and verify an interface between Rhapsody and STK to support interoperability between a SysML model in Rhapsody and an analysis model in STK. MIM may also provide guidance as to how the models are developed in both tools to support their interoperability.

From a broader perspective, MIM addresses higher-level objectives including capturing, managing, and reusing modeling & simulation (M&S) knowledge (e.g., modeling intent), achieving greater degrees of automation, increasing modularity and reusability, and so on. The "Enabling Capabilities" column in the table in Slide 1-27 identifies specific aspects that MIM aims to achieve. The "Primary Objectives" portion of this table identifies candidate ultimate objectives: reducing time, cost, and risk, as well as increasing understanding, corporate memory, and artifact (system) performance.

From this perspective one can consider modeling and simulation environments such as the excavator testbed (Figure 1) to be systems themselves. Then aspects of the table in Slide 1-27 can be viewed as potential measures of effectiveness for such systems, and similar MBSE principles can be applied. Tamburini, Peak, Paredis et al. [2005] took one of the initial steps towards this end and identified 12 capabilities, 15 challenges, 31 use cases, and 46 requirements for such environments.

Structural aspects

One main facet of MIM is its structure, which deals with what kinds of things are included in MIM and what are their relationships to each other. Specific structural aspects which may be elaborated in future work include:

- Describing emerging patterns based on Figure 1 (a) and (b), and associated refactoring that may be needed.
- Including a description of the two main interface/wrapper approaches that this work has manifested:
  - Approach 1: "connect to a subset of an existing model" (e.g., *XaiTools* and NX)
  - Approach 2: "fully auto-generate solver model, use it, dispose it" (e.g., *XaiTools* and Mathematica)

The need for multiple meta-levels of MIM is also becoming evident (similar to UML and MDA meta-levels):

- A concept-level MIM model (for methodologists/architects) that is abstract and normalized.
- Implementation-level MIM models (for end users) that provide convenient building blocks and ready-to-use leaf-level capabilities.

Process and workflow aspects

Another main facet of MIM deals with the processes and workflows for creating and using MIM-based environments. Here are a few such considerations that may be further developed in future work:

- Include a process model describing MIM in terms of identifying patterns, composing models, specifying interfaces, realizing interfaces, and so on. In general, describe how to start using the MIM approach in Company X (i.e., in environments beyond the excavator testbed).

- Help answer questions such as "If I want to do Y in a similar testbed/environment, what is the sequence of decisions and actions that I need to go through?" For example:
  - Check if an interface to the solver you need to use exists, if not, get it developed; if so, here is how you use it on your design problem ...
- Answer such questions at a high-level (it may help to assume an existing environment is already seeded and in place).
- Consider these main roles (types of actors) that are needed to utilize MIM, including the following (see elaboration below for SysML parametrics end users):

Roles (use case actors)	Example use cases
Template end user	Uses pre-wired c0 template regularly on new designs
SysML template/simulation author/user	Creates new type of c0 template using d0/e0
Generic building block librarians	Creates/updates/d0 libraries
Interface developers/integrators	Creates/maintains new a0-b0 i/f, new e0 i/f, ...
M&S environment system managers	Decides what/when new a0/e0 tools added, ...

  Each of these roles has associated process models that can be quite different from one another.
- It may also help to contrast with examples from other testbeds and approaches (e.g. different MIM-based approaches used for a wafer fab manufacturing simulation testbed vs. for this excavator manufacturing simulation testbed).

For example, we have found it helpful to differentiate among several types of end users who work with SysML parametrics as follows (e.g, with specific pointers for people using MagicDraw and ParaMagic in this case). This differentiation also helps decompose a complex topic area and define levels of learning that can be incrementally achieved.

- *Type 1 User:* Someone who works with an existing c0 model template (including executing it and performing additional instance-oriented interactions such as solving, modifying values, changing causalities, and re-solving).
  - This type of user needs the least amount of SysML and parametrics know-how.
  - This a good place to start for the casual user, for someone wanting to do basic demos, or someone just beginning to explore SysML parametrics.
- *Type 2 User:* Someone who modifies the structure of an existing c0 model template and/or creates new instances.
  - This type of user requires more know-how.
  - They also need a fair amount of MagicDraw SysML tool-aided modification support (in some respects more than Type 3 users, because they have to know how to migrate existing models and keep things consistent).
  - This is a good step towards becoming a Type 3 user.
- *Type 3 User:* Someone who creates their own c0 model structures and instances from scratch (and/or from pre-existing building blocks from a library). May drive need for new/updated d0 and a0/b0 capabilities.
  - This type of user requires a fair amount of know-how and needs good MagicDraw SysML support.
- *Type 4 User:* Someone who creates d0 building block libraries that Type 2 and Type 3 users can utilize (which may also drive interfaces to new e0 tools).

- This type of user requires similar skills as Type 3, but with a bent towards making their work reusable and modular, as well as providing good documentation and rigorous validation.

Here is a specific example illustrating these user types in for SysML parametric models (a subset of technology employed by MIM): the First Pass section in the ParaMagic Users Guide provides a quick introduction for Type 1 users. After completing the First Pass, users can work at the Type 1 level with all the pre-built tutorial models and examples provided in ParaMagic. After completing the First Pass, they also have a good "big picture" basis to proceed with the step-by-step ParaMagic tutorials. After completing the tutorials, they have achieved a good foundation to work as a Type 3 user. Becoming a Type 4 user requires different skills and know-how.

VV&A applications

At the beginning of each concept section above (CT-xx) we have indicated which MIM patterns are related to that concept (and also in the text by [MIM pattern yy] markings). Thus, VV&A use cases can be considered as one subset of use cases that the MIM architecture facilitates. In other words, the comprehensive and holistic infrastructure enabled by MIM as described above provides a rich SysML-enabled context for VV&A as well as other system lifecycle activities.

CT-17 Other Concept Extensions

Given the above concepts and examples as a basis, this section describes other concepts that we believe can be readily supported via similar means. We recommend future project phases to help explore and demonstrate these capabilities via specific test cases.

CT-17.1 Auto-generating documents from SysML models to support VV&A

SysML authoring tools like MagicDraw typically support various means for auto-generating reports and other documents based on model content. Zamparelli and Karban [2010] describe this approach for various documents needed by traditional stakeholders in the development of large-scale ground-based telescopes. Cole et al. [2010] highlight a related approach for space systems specifications and other documents. This "model-based documentation" technology can be similarly applied to aid VV&A for M&S, including:

- Generating verification traceability and status reports.
- Generating validation traceability and status reports.
- Generating accreditation documents.

CT-17.2 Managing accreditation workflows and artifacts

Building on the ideas in CT-17.1, SysML can also help in formalizing and managing workflows (and related artifacts) like those in the M&S accreditation process. NASA JPL did something like this[5] for processes in the Constellation program to effectively accredit that related systems would perform as expected. By implementing this capability as a tool plugin in MagicDraw SysML tool, they were able not only to passively visualize the workflow, but also to actively manage and re-define workflows including updating activity status and artifact issues. While their application was different than the DoD M&S accreditation process, it demonstrated the applied technologies for such problems and the resulting value and benefits.

CT-17.3 Aiding M&S validation via test results data capture and comparator usage

One type of M&S validation process is running tests on physical systems and then ensuring that the simulation re-produces similar results to a reasonable degree. SysML can help with at least some aspects of this in the case of quantitative measurements. For example, if you have a model that predicts the internal steady-state temperatures of communications gear under various operating conditions, you can run experiments on physical specimens and capture those measurements in a SysML model. You can wrap your model using a CT-13-like concept [MIM pattern e0] and then apply comparator building blocks (CT-6) to automatically compare your model results with the measured results and auto-verify if the deltas conform to acceptable limit requirements (using extensions to CT-8 and applying CT-7).

CT-17.4 Capturing SME validation criteria for future automated re-validation usage

Another type of M&S validation process is having subject matter experts (SMEs) evaluate your simulation by putting it through its paces, looking for holes, and pushing it to its limits (i.e., trying to break your simulation). At least some aspects of what these SMEs do is this: they put the simulation into a known state and then verify that the simulation responds in an expected way (e.g., see the constant equatorial orbit example in CT-14.3 for an STK-based simulation, where resulting ground station linkage durations are then expected to be constant).

When you recognize and capture these types of SME validation tests, they then effectively become verification tests that you can run own your own thereafter. Thus, you can then employ the appropriate auto-verification concept (from CT-7 through CT-15) and create an automated "virtual SME" validation suite. Thereafter whenever you change your simulation (e.g, before releasing a new version) you can ensure it passes this validation suite.

Of course this does not replace the need for actual SME-based validation processes, as SMEs will surely do new things that you did not capture (over time you can capture more of what they do and enhance your "virtual SME" validation suite), as well as do some things that are difficult if not impossible to quantify and capture. But with this approach at least you can capture and automate some SME validation aspects and thereby catch related issues earlier and more effectively.

[5] Systems Engineering Process for Operations Definition (SEPOD) Independent Analysis Team (IAT) tool for managing Mission Operations Architecture Description Document (MO ADD) generation and review (as seen via demos and personal communications Feb 2010).

CT-17.5 Managing simulation data flow and data pedigree (for sim inputs/outputs)

Some simulation environments involve a complex network of data flows in-to and out-of various sources and tools that can span multiple organizational boundaries (e.g., in the NASA GRAIL[6] project that involves multiple NASA centers and contractors). Sometimes these flows are not well-understood or well-documented, thus making it difficult to VV&A the simulations involved.

SysML can help at a basic level by treating the simulation environment as a system and simply documenting its flows in a formalized manner (e.g., by using SysML block, flow, parametrics, and activity constructs). This alone can provide key benefits including improved data pedigree and version control. A more sophisticated level is using the SysML model not just as documentation but as the means to execute and control the simulation tool chain (similar to the SysML parametrics and activity examples above in CT-13 and CT-15).

CT-17.6 Managing models & simulations themselves as systems using SysML (with requirements, structure, behavior, etc.)

Building on the previous concept (CT-17.5), you can consider an M&S environment itself (and its components) to be a system composed of subsystems and building blocks. You can then apply systems engineering concepts to M&S itself (including requirements, structure, and behavior) and manage the M&S lifecycle (including VV&A processes) using SysML-based MBSE/MBE technology (and reap similar benefits).

[6] http://science.nasa.gov/missions/grail/

4 Summary & Recommendations

The above 17+ main concepts and associated examples provide a sampler of VV&A use cases along multiple system dimensions including system levels, tools, methods, and lifecycle phases. We leveraged software V&V concepts like those seen in JUnit and showed how to create and utilize fundamental building blocks to make V&V for M&S more iterative and ubiquitous throughout the system lifecycle. Altogether this evidence demonstrates how a SysML and MBSE approach is highly capable in terms of both breadth and depth.

In summary we achieved the primary objective of this quick-look Phase 1 project (Slide 4-2):

- We showed how VV&A can be made more model-based, more embedded, and more automated so that VV&A can occur more incrementally and iteratively throughout the lifecycle (vs. after-the-fact as is often done today in a document-based checklist manner).

Along the way we also achieved these supporting secondary objectives:

- Applied known M&S patterns and developed new patterns where needed.
- Demonstrated the approach by extending existing testbeds and examples.
- Provided a basis for developing DoD-specific testbeds and deploying technology for DoD programs in future phases. See the recommended Deployment Roadmap in Slides 4-5 through 4-7.

We recommend these next project phases to accelerate progress along the roadmap:

- *Phase 2 (proposed for FY11):* Demonstrate SysML-based VV&A approach in DoD-relevant prototype testbeds. In particular, apply this approach to develop a SysML-based architecture for a current modeling activity of interest to the sponsor such as under body blast (UBB) or similar efforts. In parallel pursue technical advancements per research needs seen in Slide 4-8.
- *Phase 3+ (proposed for FY12+):* Develop multi-language/multi-technology ecosystem architecture combining AADL & SysML capabilities plus related diverse tools as examples

The main benefits we envision from this proposed multi-phase effort and subsequent deployments are the following (see also Slide 4-4):

- Enable significant improvements via new methods that make VV&A much more embedded and automated throughout the lifecycle, thereby:
  - Increasing knowledge capture and completeness.
  - Increasing modularity and reusability.
  - Increasing traceability.
  - Reducing manual effort and associated errors.
  - Increasing automation and consistency.
- Utilize these technical capabilities to reduce cost/time/risk and increase understanding, corporate memory, and system performance.
- Provide examples and reference implementations so that the DoD supply chain can begin to learn and apply these approaches.
- Provide demos of how the philosophy and techniques could be transferred and utilized to impact the DoD acquisition lifecycle.

REFERENCES

Georgia Tech MBSE/SysML-related resources

- http://www.pslm.gatech.edu/projects/incose-mbse-msi/
- http://www.pslm.gatech.edu/topics/sysml
- http://www.pslm.gatech.edu/courses

Bajaj M, Peak RS, Paredis CJJ (2007) Knowledge Composition for Efficient Analysis Problem Formulation, Part 1: Motivation and Requirements. DETC2007-35049, Proc ASME CIE Intl Conf, Las Vegas.

Bajaj M, Peak RS, Paredis CJJ (2007) Knowledge Composition for Efficient Analysis Problem Formulation, Part 2: Approach and Analysis Meta-Model. DETC2007-35050, Proc ASME CIE Intl Conf, Las Vegas.

Cole B, Delp C, Donahue K (2010) Piloting Model-Based Engineering Techniques for Spacecraft Concepts. INCOSE International Symposium, Chicago. Received *Best Paper Award*.

OMG Certified Systems Modeling Professional (OCSMP) program website: www.omg.org/ocsmp

OMG SysML™ website: www.omgsysml.org

Peak RS, Fulton RE, Nishigaki I, Okamoto N (1998) Integrating Engineering Design and Analysis Using a Multi-Representation Approach. *Engineering w. Computers*, 14 (2) 93-114.

Peak RS, Burkhart RM, Friedenthal SA, Wilson MW, Bajaj M, Kim I (2007) Simulation-Based Design Using SysML—Part 1: A Parametrics Primer. INCOSE Intl. Symposium, San Diego.

Peak RS, Burkhart RM, Friedenthal SA, Wilson MW, Bajaj M, Kim I (2007) Simulation-Based Design Using SysML—Part 2: Celebrating Diversity by Example. INCOSE Intl. Symposium, San Diego.

Peak RS, CJJ Paredis, LF McGinnis (2009-04) Model-Based SE Using SysML. NDIA Modeling and Simulation (M&S) Committee Meeting, Arlington, Virginia. Part 1: Integrating Design and Assessment M&S. Part 2: Integrating Manufacturing Design and Simulation.

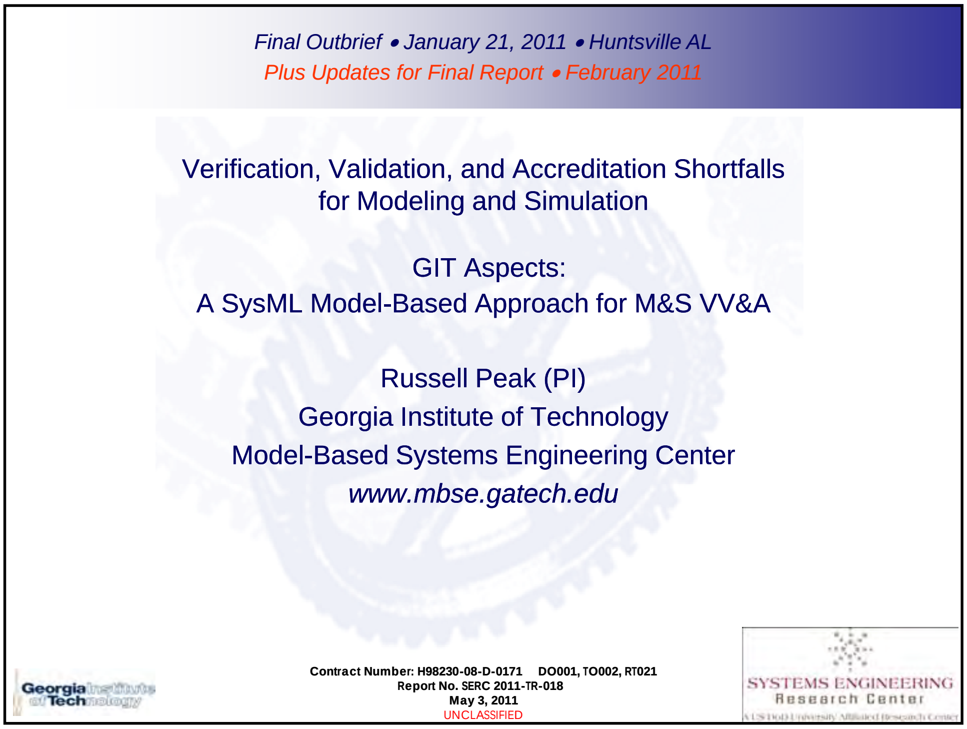
Peak RS, CJJ Paredis, LF McGinnis, SA Friedenthal, RM Burkhart, et al. (2009-11) Integrating System Design with Simulation and Analysis Using SysML. INCOSE MBSE Challenge, Modeling & Simulation Interoperability (MSI) Team, Phase 2 Final Report (v2.0). http://www.pslm.gatech.edu/projects/incose-mbse-msi/

Peak RS, Paredis CP, McGinnis LF, Friedenthal SA, Burkhart RM (2009-12) Integrating System Design with Simulation and Analysis Using SysML. *Special Issue: Model-Based Systems Engineering: The New Paradigm, INCOSE Insight* (12) 4.

Peak RS, CJJ Paredis, LF McGinnis, SA Friedenthal, RM Burkhart, M Bajaj (2010-02) INCOSE Model-Based Systems Engineering (MBSE) Challenge: Modeling & Simulation Interoperability (MSI) Team Status Update [With Applications to Mechatronics, Other Cyber-Physical Systems, and Beyond]. INCOSE Intl Workshop, Mesa AZ.

Tamburini DR, Peak RS, Paredis CJJ (2005-10) Composable Objects (COB) Requirements & Objectives v1.0. Technical Report, Georgia Tech, Atlanta. http://eislab.gatech.edu/projects/nasa-ngcobs/

M Zamparelli and R Karban (2010-12) Model Based Document Generation. INCOSE MBSE Initiative webinar. http://www.omgwiki.org/MBSE/doku.php?id=mbse:telescope



Final Outbrief • January 21, 2011 • Huntsville AL

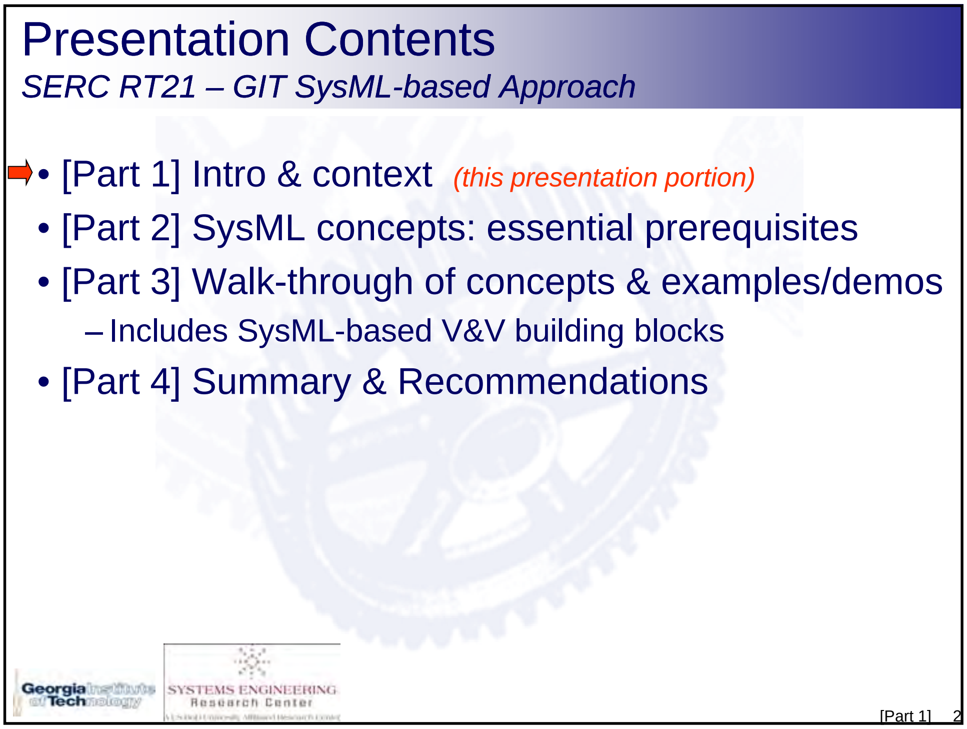
Plus Updates for Final Report • February 2011

Verification, Validation, and Accreditation Shortfalls
for Modeling and Simulation

GIT Aspects:
A SysML Model-Based Approach for M&S VV&A

Russell Peak (PI)
Georgia Institute of Technology
Model-Based Systems Engineering Center
www.mbse.gatech.edu

Contract Number: H98230-08-D-0171 DO001, TO002, RT021
Report No. SERC 2011-TR-018
May 3, 2011
UNCLASSIFIED

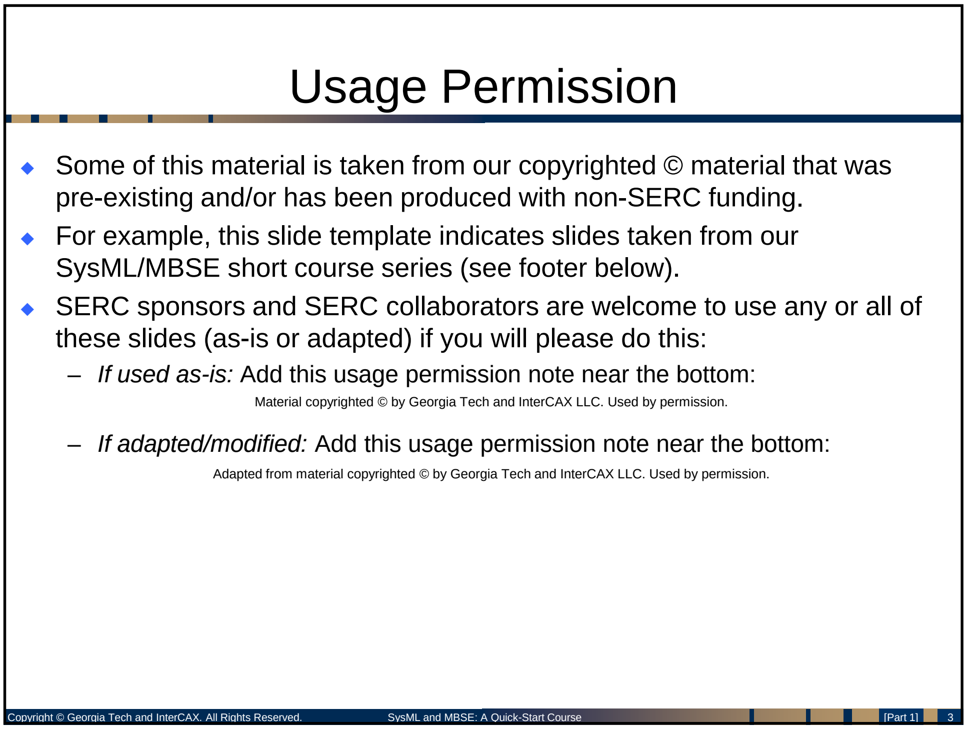


Presentation Contents

SERC RT21 – GIT SysML-based Approach

- [Part 1] Intro & context *(this presentation portion)*
- [Part 2] SysML concepts: essential prerequisites
- [Part 3] Walk-through of concepts & examples/demos
  - Includes SysML-based V&V building blocks
- [Part 4] Summary & Recommendations

[Part 1] 2

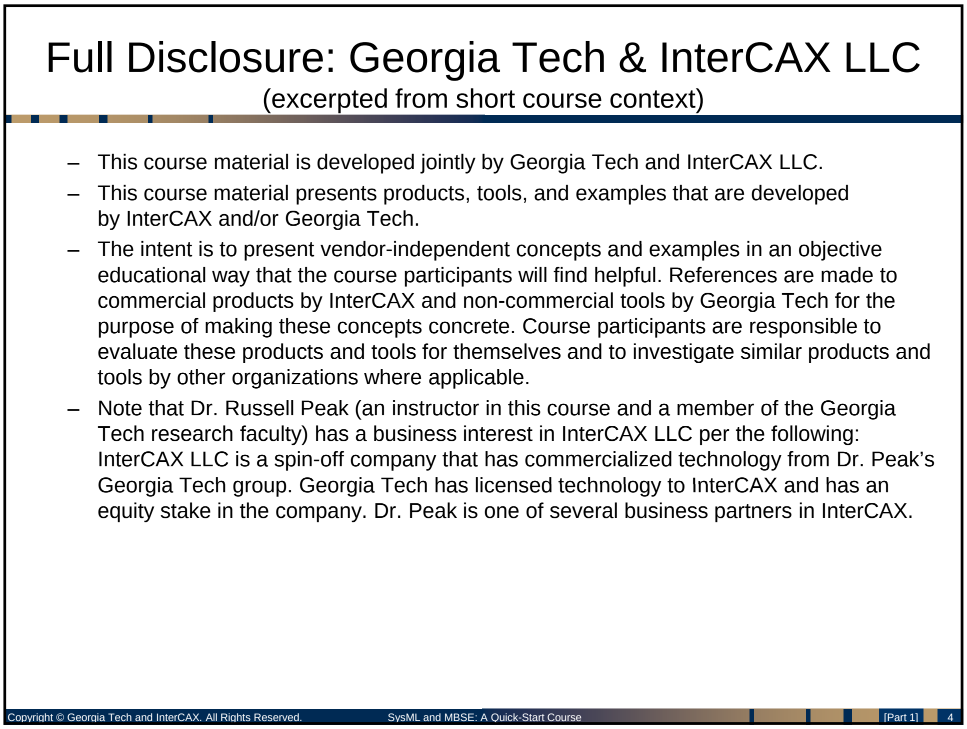


Usage Permission

- Some of this material is taken from our copyrighted © material that was pre-existing and/or has been produced with non-SERC funding.
- For example, this slide template indicates slides taken from our SysML/MBSE short course series (see footer below).
- SERC sponsors and SERC collaborators are welcome to use any or all of these slides (as-is or adapted) if you will please do this:
  - *If used as-is:* Add this usage permission note near the bottom:

    Material copyrighted © by Georgia Tech and InterCAX LLC. Used by permission.
  - *If adapted/modified:* Add this usage permission note near the bottom:

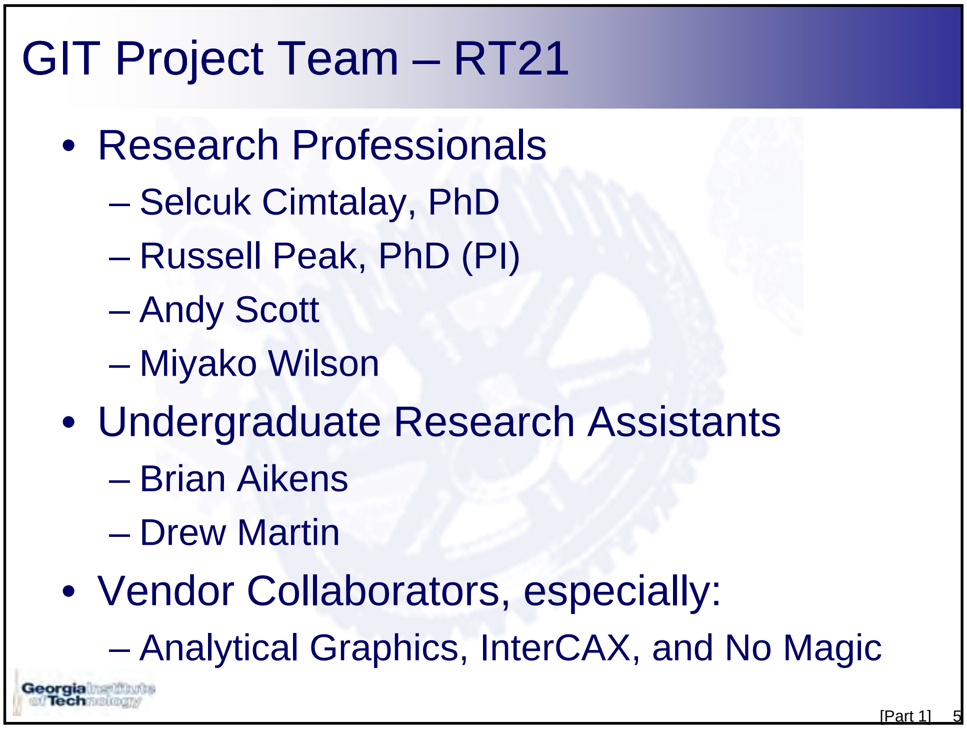
    Adapted from material copyrighted © by Georgia Tech and InterCAX LLC. Used by permission.



Full Disclosure: Georgia Tech & InterCAX LLC

(excerpted from short course context)

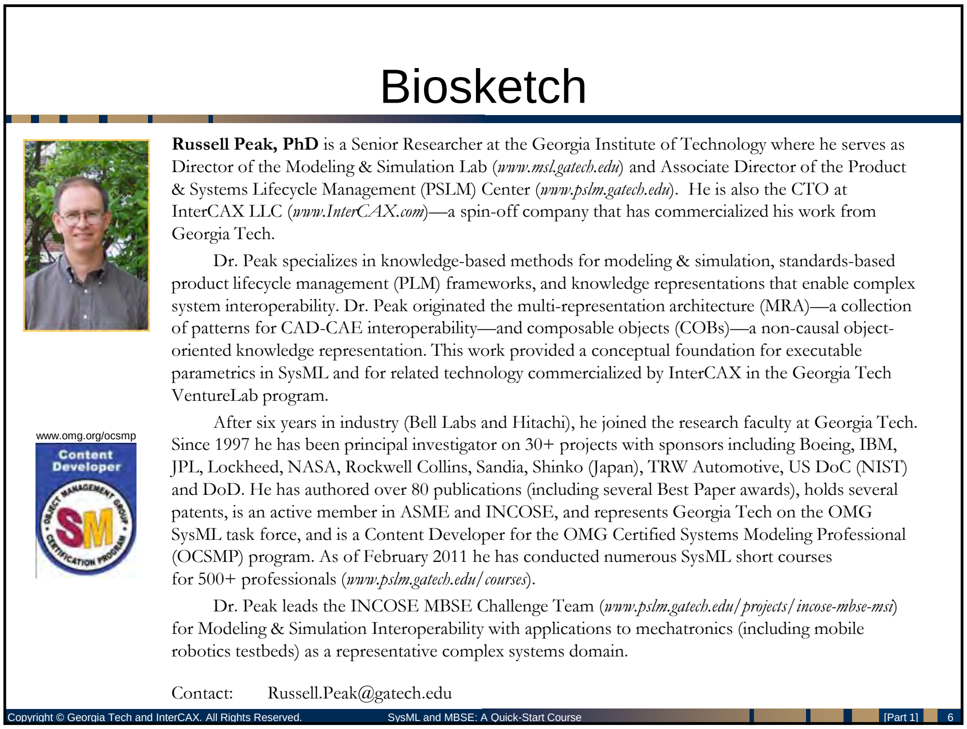
- This course material is developed jointly by Georgia Tech and InterCAX LLC.
- This course material presents products, tools, and examples that are developed by InterCAX and/or Georgia Tech.
- The intent is to present vendor-independent concepts and examples in an objective educational way that the course participants will find helpful. References are made to commercial products by InterCAX and non-commercial tools by Georgia Tech for the purpose of making these concepts concrete. Course participants are responsible to evaluate these products and tools for themselves and to investigate similar products and tools by other organizations where applicable.
- Note that Dr. Russell Peak (an instructor in this course and a member of the Georgia Tech research faculty) has a business interest in InterCAX LLC per the following: InterCAX LLC is a spin-off company that has commercialized technology from Dr. Peak's Georgia Tech group. Georgia Tech has licensed technology to InterCAX and has an equity stake in the company. Dr. Peak is one of several business partners in InterCAX.



GIT Project Team – RT21

- Research Professionals
  - Selcuk Cimtalay, PhD
  - Russell Peak, PhD (PI)
  - Andy Scott
  - Miyako Wilson
- Undergraduate Research Assistants
  - Brian Aikens
  - Drew Martin
- Vendor Collaborators, especially:
  - Analytical Graphics, InterCAX, and No Magic

Georgia Institute of Technology



Biosketch

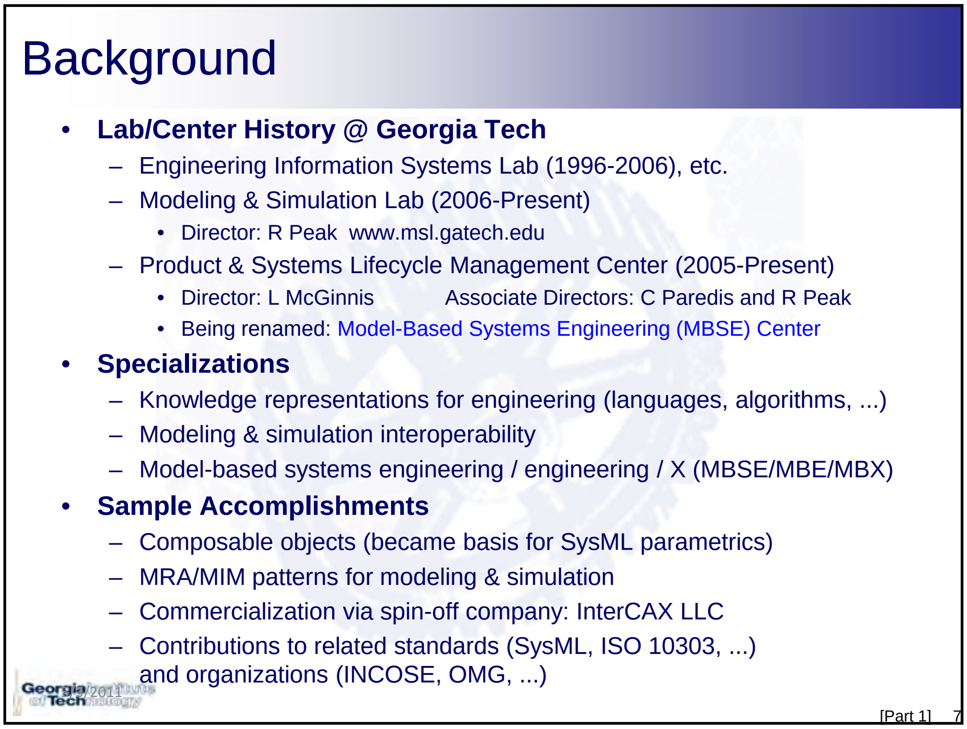
Russell Peak, PhD is a Senior Researcher at the Georgia Institute of Technology where he serves as Director of the Modeling & Simulation Lab (*www.msl.gatech.edu*) and Associate Director of the Product & Systems Lifecycle Management (PSLM) Center (*www.pslm.gatech.edu*). He is also the CTO at InterCAX LLC (*www.InterCAX.com*)—a spin-off company that has commercialized his work from Georgia Tech.

Dr. Peak specializes in knowledge-based methods for modeling & simulation, standards-based product lifecycle management (PLM) frameworks, and knowledge representations that enable complex system interoperability. Dr. Peak originated the multi-representation architecture (MRA)—a collection of patterns for CAD-CAE interoperability—and composable objects (COBs)—a non-causal object-oriented knowledge representation. This work provided a conceptual foundation for executable parametrics in SysML and for related technology commercialized by InterCAX in the Georgia Tech VentureLab program.

After six years in industry (Bell Labs and Hitachi), he joined the research faculty at Georgia Tech. Since 1997 he has been principal investigator on 30+ projects with sponsors including Boeing, IBM, JPL, Lockheed, NASA, Rockwell Collins, Sandia, Shinko (Japan), TRW Automotive, US DoC (NIST) and DoD. He has authored over 80 publications (including several Best Paper awards), holds several patents, is an active member in ASME and INCOSE, and represents Georgia Tech on the OMG SysML task force, and is a Content Developer for the OMG Certified Systems Modeling Professional (OCSMP) program. As of February 2011 he has conducted numerous SysML short courses for 500+ professionals (*www.pslm.gatech.edu/courses*).

Dr. Peak leads the INCOSE MBSE Challenge Team (*www.pslm.gatech.edu/projects/incose-mbse-msi*) for Modeling & Simulation Interoperability with applications to mechatronics (including mobile robotics testbeds) as a representative complex systems domain.

Contact: Russell.Peak@gatech.edu

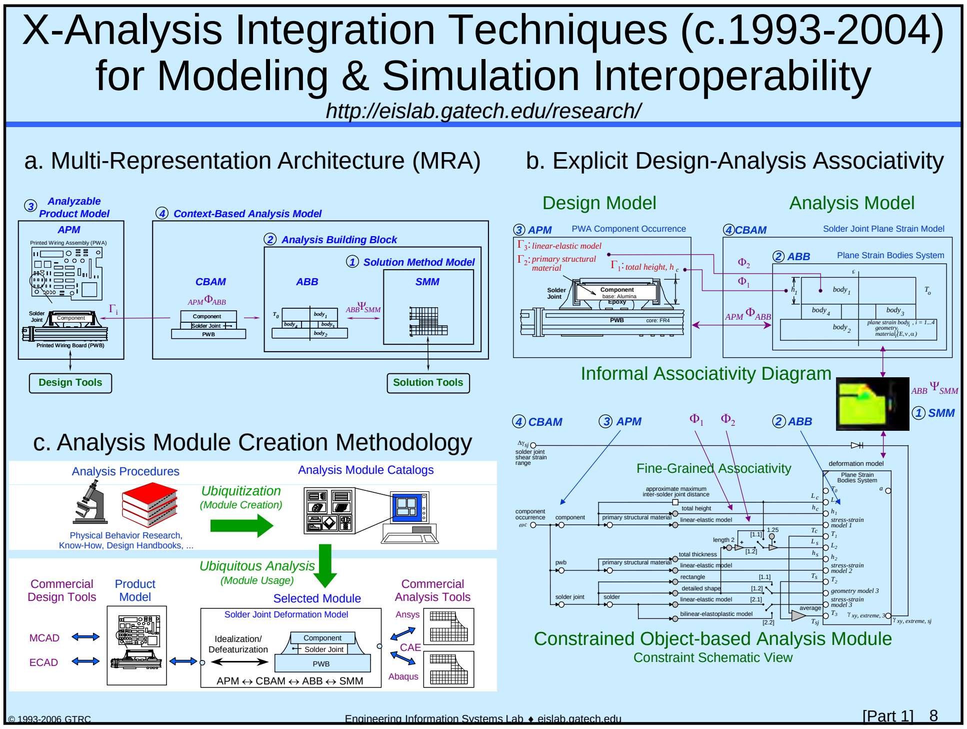


Background

- **Lab/Center History @ Georgia Tech**
  - Engineering Information Systems Lab (1996-2006), etc.
  - Modeling & Simulation Lab (2006-Present)
    - Director: R Peak www.msl.gatech.edu
  - Product & Systems Lifecycle Management Center (2005-Present)
    - Director: L McGinnis Associate Directors: C Paredis and R Peak
    - Being renamed: Model-Based Systems Engineering (MBSE) Center
- **Specializations**
  - Knowledge representations for engineering (languages, algorithms, ...)
  - Modeling & simulation interoperability
  - Model-based systems engineering / engineering / X (MBSE/MBE/MBX)
- **Sample Accomplishments**
  - Composable objects (became basis for SysML parametrics)
  - MRA/MIM patterns for modeling & simulation
  - Commercialization via spin-off company: InterCAX LLC
  - Contributions to related standards (SysML, ISO 10303, ...) and organizations (INCOSE, OMG, ...)

Georgia Institute of Technology ©2011

[Part 1] 7



X-Analysis Integration Techniques (c.1993-2004) for Modeling & Simulation Interoperability

http://eislab.gatech.edu/research/

a. Multi-Representation Architecture (MRA)

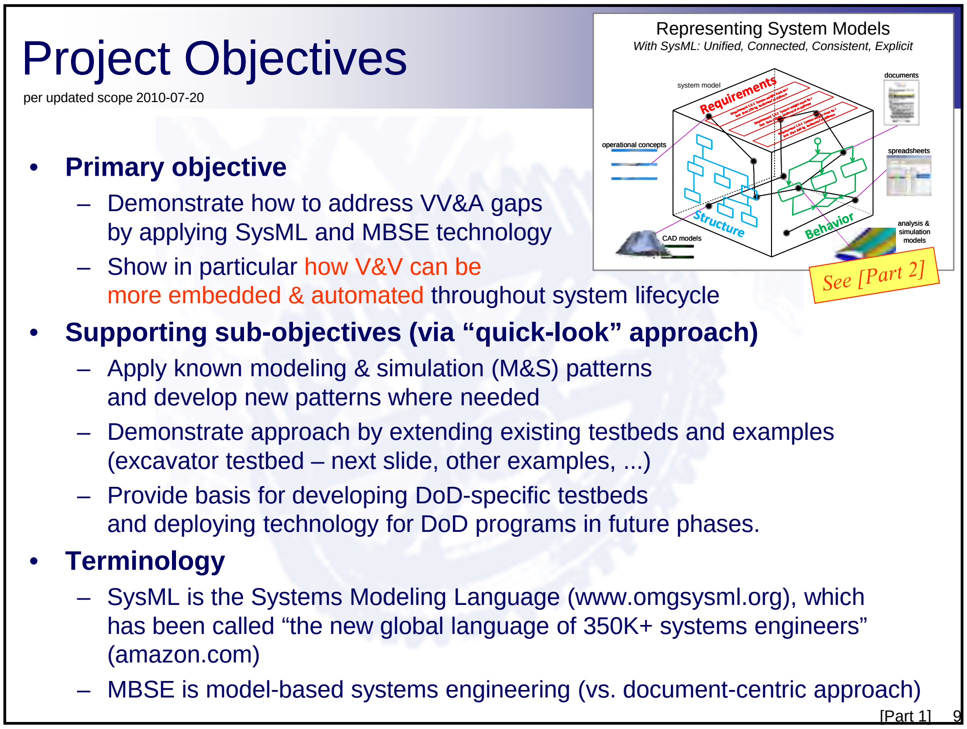
b. Explicit Design-Analysis Associativity

Constrained Object-based Analysis Module

Constraint Schematic View

c. Analysis Module Creation Methodology

© 1993-2006 GTRC Engineering Information Systems Lab ♦ eislab.gatech.edu [Part 1] 8

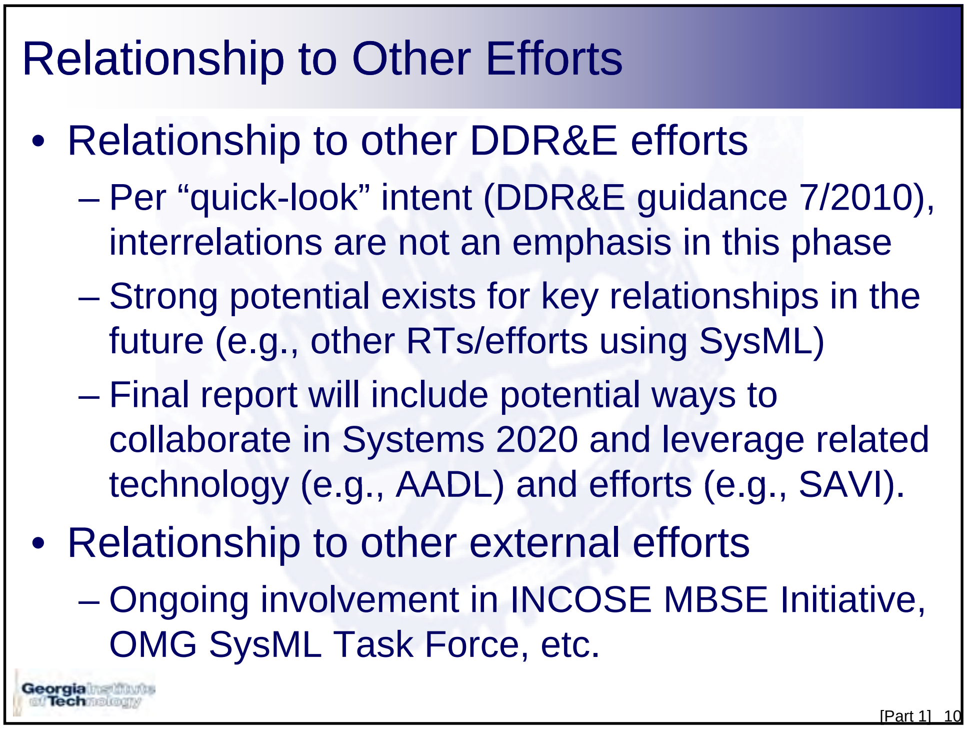


Project Objectives

per updated scope 2010-07-20

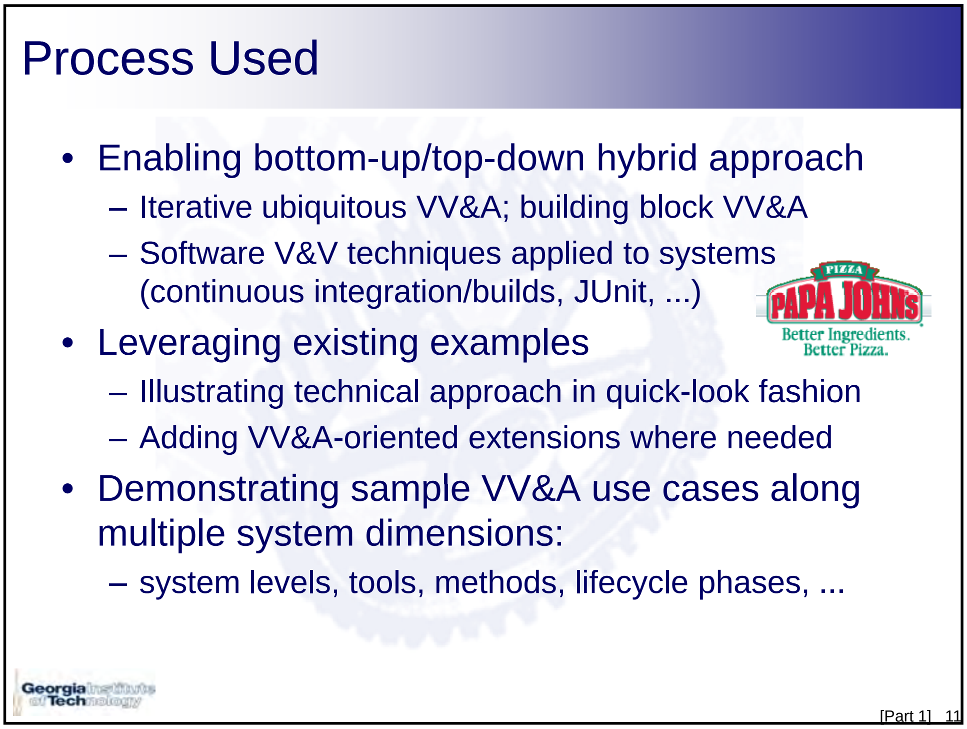
Representing System Models
With SysML: Unified, Connected, Consistent, Explicit
system model
Requirements
Structure
Behavior
documents
operational concepts
spreadsheets
CAD models
analysis & simulation models
See [Part 2]

- **Primary objective**
  - Demonstrate how to address VV&A gaps by applying SysML and MBSE technology
  - Show in particular how V&V can be more embedded & automated throughout system lifecycle
- **Supporting sub-objectives (via “quick-look” approach)**
  - Apply known modeling & simulation (M&S) patterns and develop new patterns where needed
  - Demonstrate approach by extending existing testbeds and examples (excavator testbed – next slide, other examples, ...)
  - Provide basis for developing DoD-specific testbeds and deploying technology for DoD programs in future phases.
- **Terminology**
  - SysML is the Systems Modeling Language (www.omgsysml.org), which has been called “the new global language of 350K+ systems engineers” (amazon.com)
  - MBSE is model-based systems engineering (vs. document-centric approach)



Relationship to Other Efforts

- Relationship to other DDR&E efforts
  - Per “quick-look” intent (DDR&E guidance 7/2010), interrelations are not an emphasis in this phase
  - Strong potential exists for key relationships in the future (e.g., other RTs/efforts using SysML)
  - Final report will include potential ways to collaborate in Systems 2020 and leverage related technology (e.g., AADL) and efforts (e.g., SAVI).
- Relationship to other external efforts
  - Ongoing involvement in INCOSE MBSE Initiative, OMG SysML Task Force, etc.



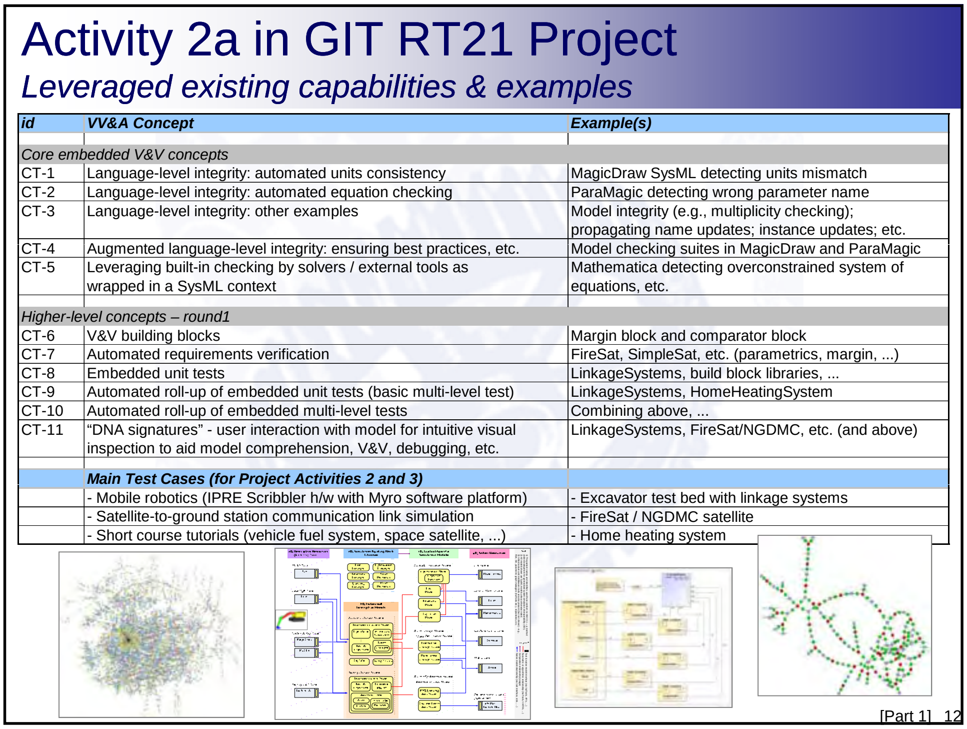
Process Used

- Enabling bottom-up/top-down hybrid approach
  - – Iterative ubiquitous VV&A; building block VV&A
  - – Software V&V techniques applied to systems (continuous integration/builds, JUnit, ...)
- Leveraging existing examples
  - – Illustrating technical approach in quick-look fashion
  - – Adding VV&A-oriented extensions where needed
- Demonstrating sample VV&A use cases along multiple system dimensions:
  - – system levels, tools, methods, lifecycle phases, ...

PIZZA PAPA JOHN'S Better Ingredients. Better Pizza.

Georgia Institute of Technology

[Part 1] 11

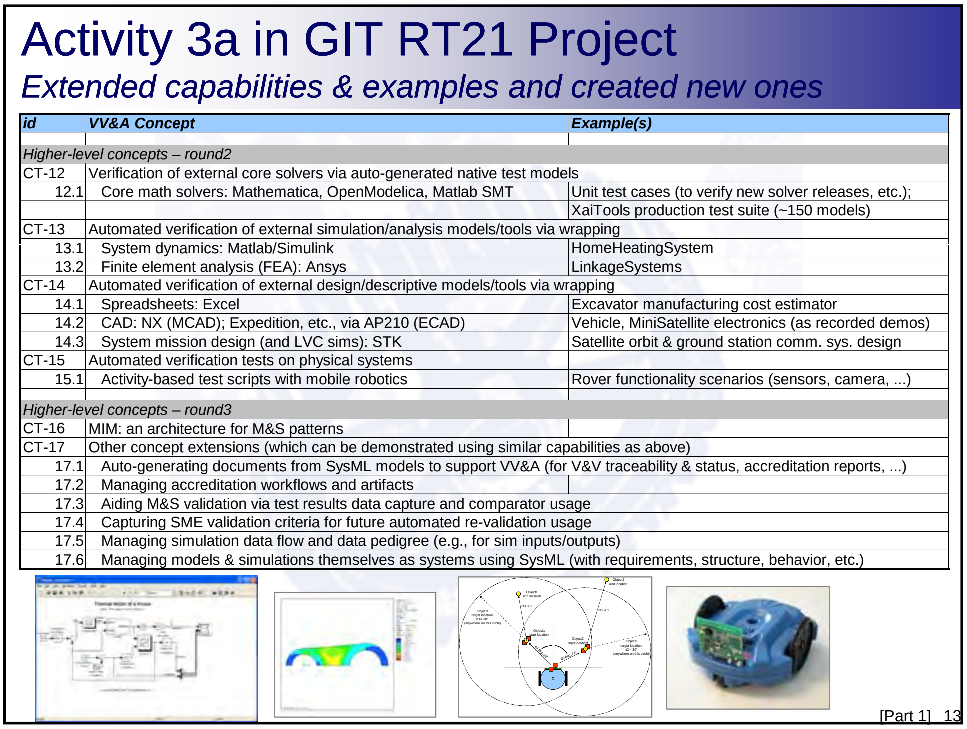


Activity 2a in GIT RT21 Project

Leveraged existing capabilities & examples

id	*VV&A Concept*	*Example(s)*
Core embedded V&V concepts		
CT-1	Language-level integrity: automated units consistency	MagicDraw SysML detecting units mismatch
CT-2	Language-level integrity: automated equation checking	ParaMagic detecting wrong parameter name
CT-3	Language-level integrity: other examples	Model integrity (e.g., multiplicity checking); propagating name updates; instance updates; etc.
CT-4	Augmented language-level integrity: ensuring best practices, etc.	Model checking suites in MagicDraw and ParaMagic
CT-5	Leveraging built-in checking by solvers / external tools as wrapped in a SysML context	Mathematica detecting overconstrained system of equations, etc.
Higher-level concepts – round1		
CT-6	V&V building blocks	Margin block and comparator block
CT-7	Automated requirements verification	FireSat, SimpleSat, etc. (parametrics, margin, ...)
CT-8	Embedded unit tests	LinkageSystems, build block libraries, ...
CT-9	Automated roll-up of embedded unit tests (basic multi-level test)	LinkageSystems, HomeHeatingSystem
CT-10	Automated roll-up of embedded multi-level tests	Combining above, ...
CT-11	"DNA signatures" - user interaction with model for intuitive visual inspection to aid model comprehension, V&V, debugging, etc.	LinkageSystems, FireSat/NGDMC, etc. (and above)
	Main Test Cases (for Project Activities 2 and 3)	
	- Mobile robotics (IPRE Scribbler h/w with Myro software platform)	- Excavator test bed with linkage systems
	- Satellite-to-ground station communication link simulation	- FireSat / NGDMC satellite
	- Short course tutorials (vehicle fuel system, space satellite, ...)	- Home heating system

[Part 1] 12

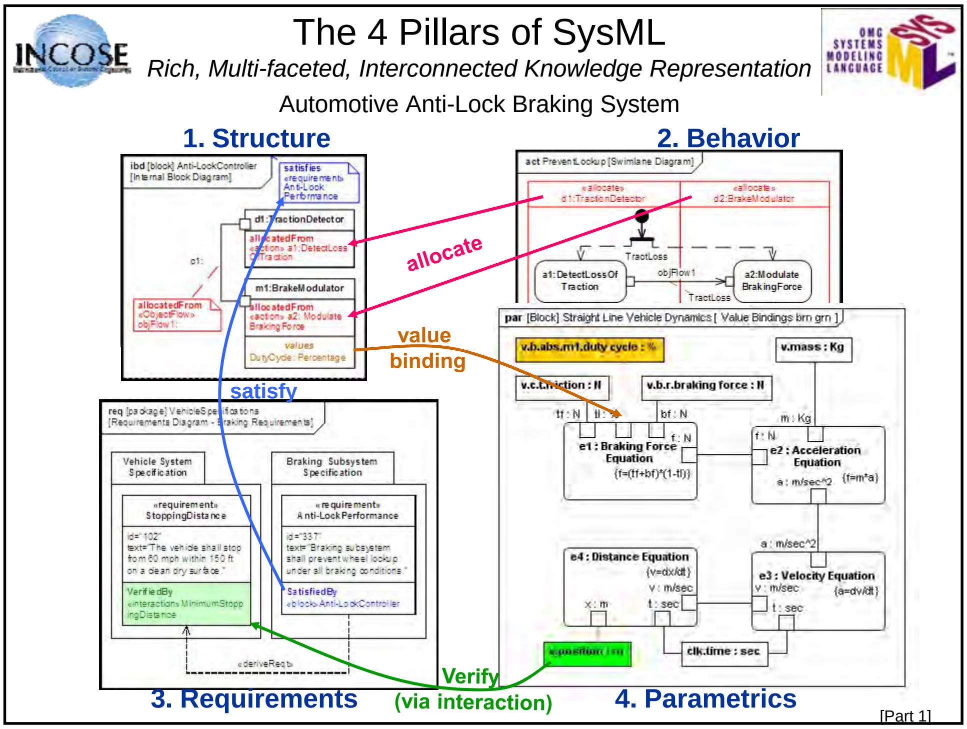


Activity 3a in GIT RT21 Project

Extended capabilities & examples and created new ones

id	*VV&A Concept*	*Example(s)*
Higher-level concepts – round2		
CT-12	Verification of external core solvers via auto-generated native test models	
12.1	Core math solvers: Mathematica, OpenModelica, Matlab SMT	Unit test cases (to verify new solver releases, etc.);
		XaiTools production test suite (~150 models)
CT-13	Automated verification of external simulation/analysis models/tools via wrapping	
13.1	System dynamics: Matlab/Simulink	HomeHeatingSystem
13.2	Finite element analysis (FEA): Ansys	LinkageSystems
CT-14	Automated verification of external design/descriptive models/tools via wrapping	
14.1	Spreadsheets: Excel	Excavator manufacturing cost estimator
14.2	CAD: NX (MCAD); Expedition, etc., via AP210 (ECAD)	Vehicle, MiniSatellite electronics (as recorded demos)
14.3	System mission design (and LVC sims): STK	Satellite orbit & ground station comm. sys. design
CT-15	Automated verification tests on physical systems	
15.1	Activity-based test scripts with mobile robotics	Rover functionality scenarios (sensors, camera, ...)
Higher-level concepts – round3		
CT-16	MIM: an architecture for M&S patterns	
CT-17	Other concept extensions (which can be demonstrated using similar capabilities as above)	
17.1	Auto-generating documents from SysML models to support VV&A (for V&V traceability & status, accreditation reports, ...)	
17.2	Managing accreditation workflows and artifacts	
17.3	Aiding M&S validation via test results data capture and comparator usage	
17.4	Capturing SME validation criteria for future automated re-validation usage	
17.5	Managing simulation data flow and data pedigree (e.g., for sim inputs/outputs)	
17.6	Managing models & simulations themselves as systems using SysML (with requirements, structure, behavior, etc.)	

[Part 1] 13



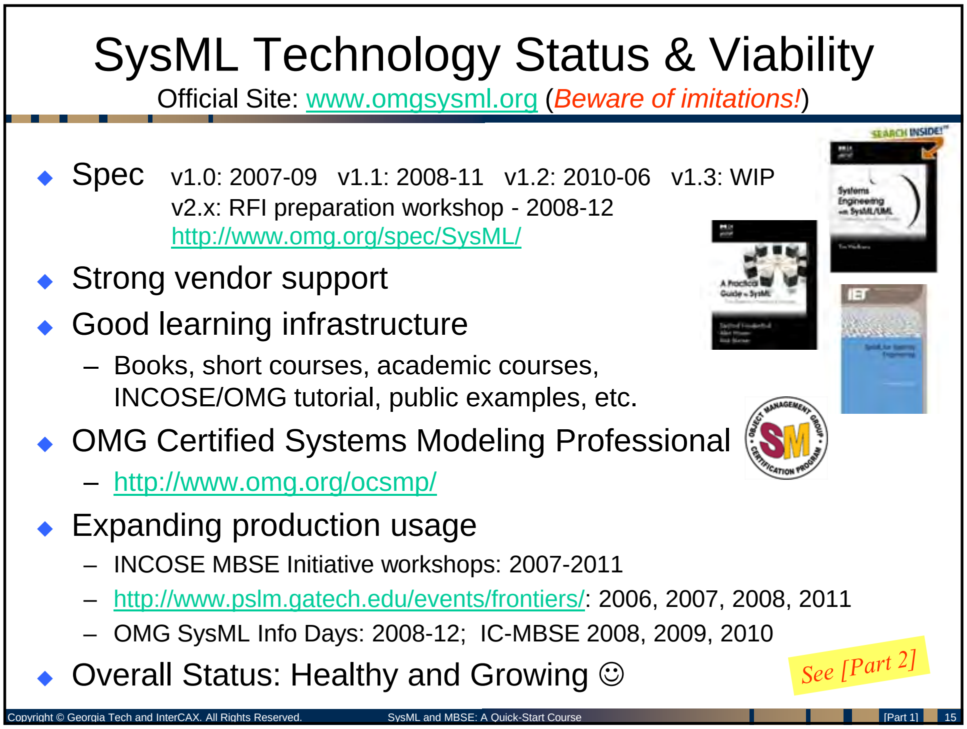
The 4 Pillars of SysML

Rich, Multi-faceted, Interconnected Knowledge Representation

Automotive Anti-Lock Braking System

1. Structure
2. Behavior
3. Requirements
4. Parametrics
allocate
value binding
satisfy
Verify (via interaction)
ibd [block] Anti-LockController [Internal Block Diagram]
satisfies «requirement» Anti-Lock Performance
d1:TractionDetector
allocatedFrom «action» a1:DetectLossOfTraction
m1:BrakeModulator
allocatedFrom «action» a2: Modulate Braking Force
values DutyCycle: Percentage
c1:
allocatedFrom «ObjectFlow» objFlow1:
act PreventLockup [Swimlane Diagram]
«allocate» d1:TractionDetector
«allocate» d2:BrakeModulator
TractLoss
a1:DetectLossOf Traction
objFlow1
TractLoss
a2:Modulate BrakingForce
req [package] VehicleSpecifications [Requirements Diagram - Braking Requirements]
Vehicle System Specification
Braking Subsystem Specification
«requirement» StoppingDistance
id="102"
text="The vehicle shall stop from 60 mph within 150 ft on a clean dry surface."
VerifiedBy «interaction» MinimumStoppingDistance
«requirement» Anti-LockPerformance
id="337"
text="Braking subsystem shall prevent wheel lockup under all braking conditions."
SatisfiedBy «block» Anti-LockController
«deriveReqt»
par [Block] Straight Line Vehicle Dynamics [Value Bindings brn grn]
v.b.abs.m1.duty cycle : %
v.mass : Kg
v.c.t.friction : N
v.b.r.braking force : N
tf : N
tl : %
bf : N
m : Kg
f : N
e1 : Braking Force Equation
{f=(tf+bf)*(1-tl)}
f : N
e2 : Acceleration Equation
a : m/sec^2
{f=m*a}
a : m/sec^2
e4 : Distance Equation
{v=dx/dt}
v : m/sec
x : m
t : sec
e3 : Velocity Equation
v : m/sec
{a=dv/dt}
t : sec
x.position : m
clk.time : sec

[Part 1]

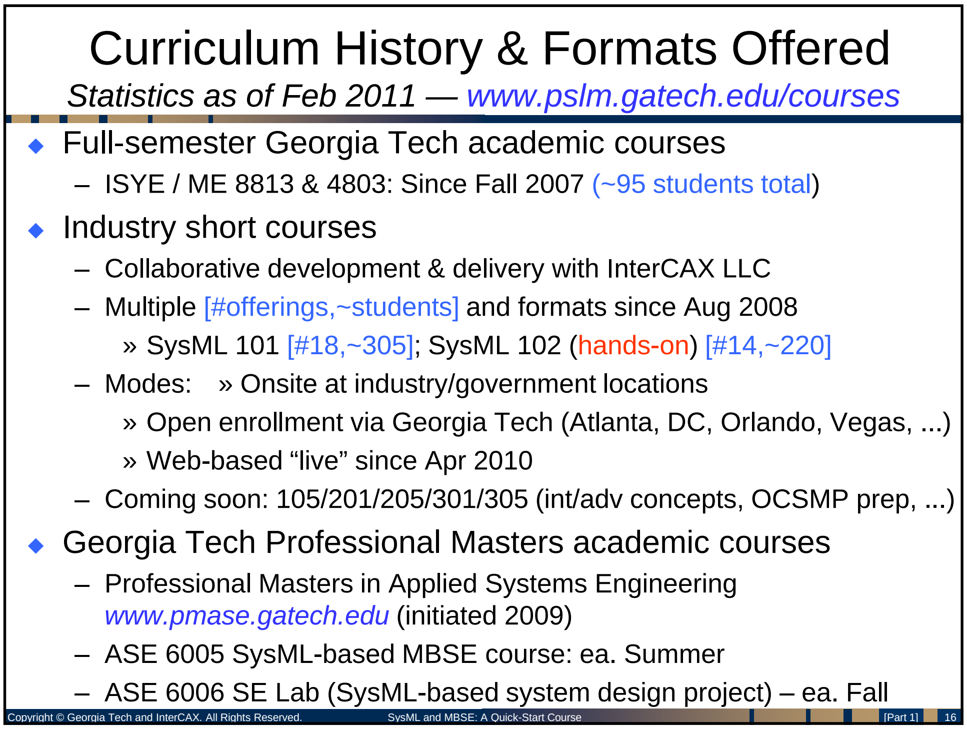


SysML Technology Status & Viability

Official Site: www.omgsysml.org (*Beware of imitations!*)

- Spec v1.0: 2007-09 v1.1: 2008-11 v1.2: 2010-06 v1.3: WIP
  v2.x: RFI preparation workshop - 2008-12
  http://www.omg.org/spec/SysML/
- Strong vendor support
- Good learning infrastructure
  - Books, short courses, academic courses, INCOSE/OMG tutorial, public examples, etc.
- OMG Certified Systems Modeling Professional
  - http://www.omg.org/ocsmp/
- Expanding production usage
  - INCOSE MBSE Initiative workshops: 2007-2011
  - http://www.pslm.gatech.edu/events/frontiers/: 2006, 2007, 2008, 2011
  - OMG SysML Info Days: 2008-12; IC-MBSE 2008, 2009, 2010
- Overall Status: Healthy and Growing ☺

See [Part 2]



Curriculum History & Formats Offered

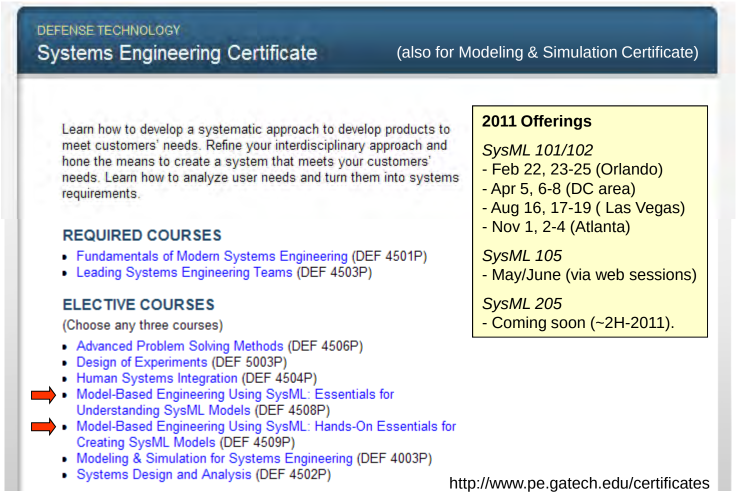
Statistics as of Feb 2011 — www.pslm.gatech.edu/courses

- Full-semester Georgia Tech academic courses
  - ISYE / ME 8813 & 4803: Since Fall 2007 (~95 students total)
- Industry short courses
  - Collaborative development & delivery with InterCAX LLC
  - Multiple [#offerings,~students] and formats since Aug 2008
    - » SysML 101 [#18,~305]; SysML 102 (hands-on) [#14,~220]
  - Modes: » Onsite at industry/government locations
    - » Open enrollment via Georgia Tech (Atlanta, DC, Orlando, Vegas, ...)
    - » Web-based “live” since Apr 2010
  - Coming soon: 105/201/205/301/305 (int/adv concepts, OCSMP prep, ...)
- Georgia Tech Professional Masters academic courses
  - Professional Masters in Applied Systems Engineering *www.pmase.gatech.edu* (initiated 2009)
  - ASE 6005 SysML-based MBSE course: ea. Summer
  - ASE 6006 SE Lab (SysML-based system design project) – ea. Fall

Open Enrollment Short Course Formats

SysML 101 (1 day), SysML 102 (2.5 days), plus others (SysML 105, etc.)

www.pslm.gatech.edu/courses



DEFENSE TECHNOLOGY

Systems Engineering Certificate

(also for Modeling & Simulation Certificate)

Learn how to develop a systematic approach to develop products to meet customers' needs. Refine your interdisciplinary approach and hone the means to create a system that meets your customers' needs. Learn how to analyze user needs and turn them into systems requirements.

REQUIRED COURSES

- Fundamentals of Modern Systems Engineering (DEF 4501P)
- Leading Systems Engineering Teams (DEF 4503P)

ELECTIVE COURSES

(Choose any three courses)

- Advanced Problem Solving Methods (DEF 4506P)
- Design of Experiments (DEF 5003P)
- Human Systems Integration (DEF 4504P)
- ➡ Model-Based Engineering Using SysML: Essentials for Understanding SysML Models (DEF 4508P)
- ➡ Model-Based Engineering Using SysML: Hands-On Essentials for Creating SysML Models (DEF 4509P)
- Modeling & Simulation for Systems Engineering (DEF 4003P)
- Systems Design and Analysis (DEF 4502P)

2011 Offerings

SysML 101/102
- Feb 22, 23-25 (Orlando)
- Apr 5, 6-8 (DC area)
- Aug 16, 17-19 (Las Vegas)
- Nov 1, 2-4 (Atlanta)

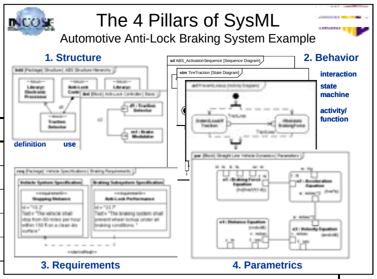
SysML 105
- May/June (via web sessions)

SysML 205
- Coming soon (~2H-2011).

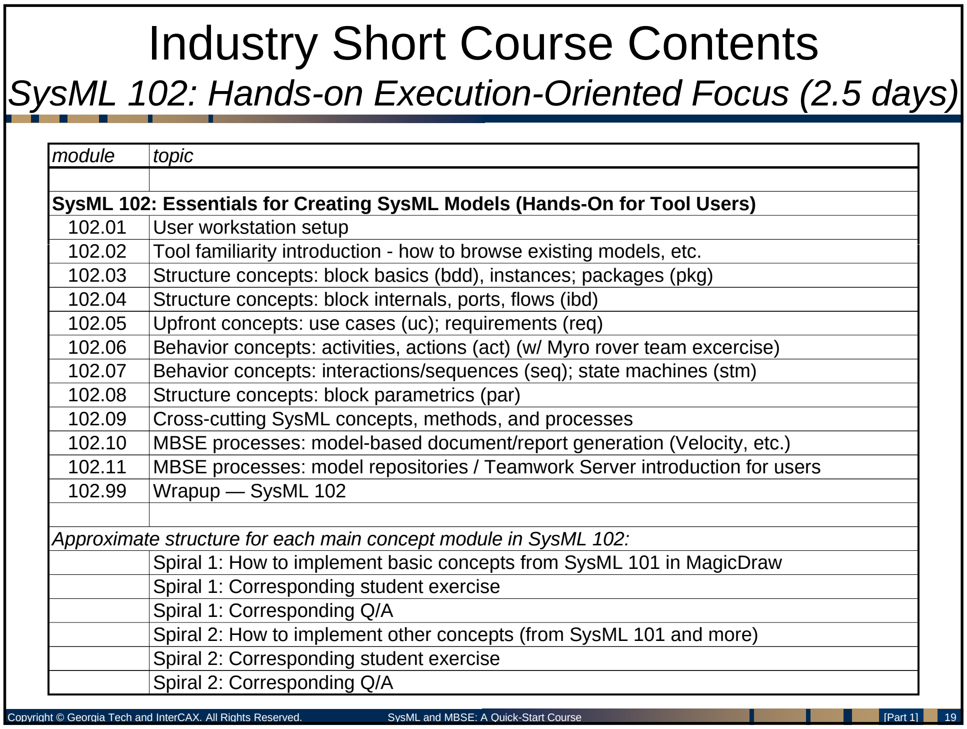
http://www.pe.gatech.edu/certificates

Industry Short Course Contents

SysML 101: Notation Comprehension Focus (1 day)



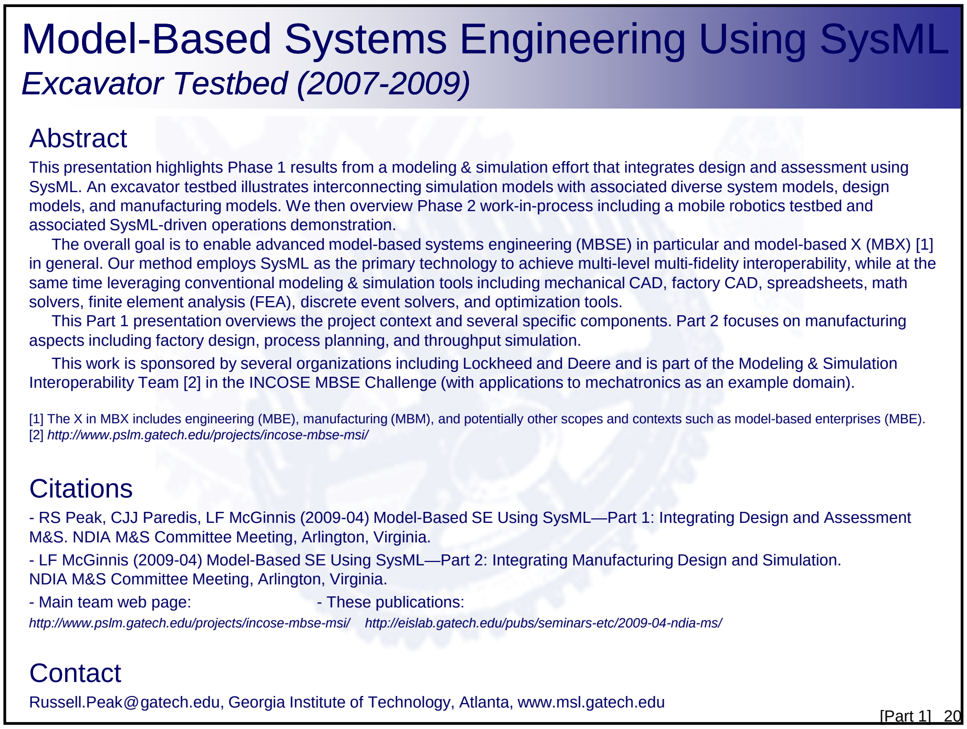
module	*topic*
Course Context	
000.01	Introduction and course overview
SysML 101: Essentials for Understanding SysML Models	
101.01	MBSE context & motivation
101.02	SysML introduction & overview; Course examples overview
101.03	Structure concepts: block basics (bdd), instances; packages (pkg)
101.04	Structure concepts: block internals, ports, flows (ibd)
101.05	Upfront concepts: use cases (uc); requirements (req)
101.06	Behavior concepts: activities, actions (act)
101.07	Behavior concepts: interactions/sequences (seq); state machines (stm)
101.08	Structure concepts: block parametrics (par)
101.09	Cross-cutting SysML concepts, methods, and processes
101.99	Wrapup — SysML 101



Industry Short Course Contents

SysML 102: Hands-on Execution-Oriented Focus (2.5 days)

module	*topic*
SysML 102: Essentials for Creating SysML Models (Hands-On for Tool Users)	
102.01	User workstation setup
102.02	Tool familiarity introduction - how to browse existing models, etc.
102.03	Structure concepts: block basics (bdd), instances; packages (pkg)
102.04	Structure concepts: block internals, ports, flows (ibd)
102.05	Upfront concepts: use cases (uc); requirements (req)
102.06	Behavior concepts: activities, actions (act) (w/ Myro rover team excercise)
102.07	Behavior concepts: interactions/sequences (seq); state machines (stm)
102.08	Structure concepts: block parametrics (par)
102.09	Cross-cutting SysML concepts, methods, and processes
102.10	MBSE processes: model-based document/report generation (Velocity, etc.)
102.11	MBSE processes: model repositories / Teamwork Server introduction for users
102.99	Wrapup — SysML 102
Approximate structure for each main concept module in SysML 102:	
	Spiral 1: How to implement basic concepts from SysML 101 in MagicDraw
	Spiral 1: Corresponding student exercise
	Spiral 1: Corresponding Q/A
	Spiral 2: How to implement other concepts (from SysML 101 and more)
	Spiral 2: Corresponding student exercise
	Spiral 2: Corresponding Q/A



Model-Based Systems Engineering Using SysML

Excavator Testbed (2007-2009)

Abstract

This presentation highlights Phase 1 results from a modeling & simulation effort that integrates design and assessment using SysML. An excavator testbed illustrates interconnecting simulation models with associated diverse system models, design models, and manufacturing models. We then overview Phase 2 work-in-process including a mobile robotics testbed and associated SysML-driven operations demonstration.

The overall goal is to enable advanced model-based systems engineering (MBSE) in particular and model-based X (MBX) [1] in general. Our method employs SysML as the primary technology to achieve multi-level multi-fidelity interoperability, while at the same time leveraging conventional modeling & simulation tools including mechanical CAD, factory CAD, spreadsheets, math solvers, finite element analysis (FEA), discrete event solvers, and optimization tools.

This Part 1 presentation overviews the project context and several specific components. Part 2 focuses on manufacturing aspects including factory design, process planning, and throughput simulation.

This work is sponsored by several organizations including Lockheed and Deere and is part of the Modeling & Simulation Interoperability Team [2] in the INCOSE MBSE Challenge (with applications to mechatronics as an example domain).

[1] The X in MBX includes engineering (MBE), manufacturing (MBM), and potentially other scopes and contexts such as model-based enterprises (MBE).
[2] *http://www.pslm.gatech.edu/projects/incose-mbse-msi/*

Citations

- RS Peak, CJJ Paredis, LF McGinnis (2009-04) Model-Based SE Using SysML—Part 1: Integrating Design and Assessment M&S. NDIA M&S Committee Meeting, Arlington, Virginia.

- LF McGinnis (2009-04) Model-Based SE Using SysML—Part 2: Integrating Manufacturing Design and Simulation. NDIA M&S Committee Meeting, Arlington, Virginia.

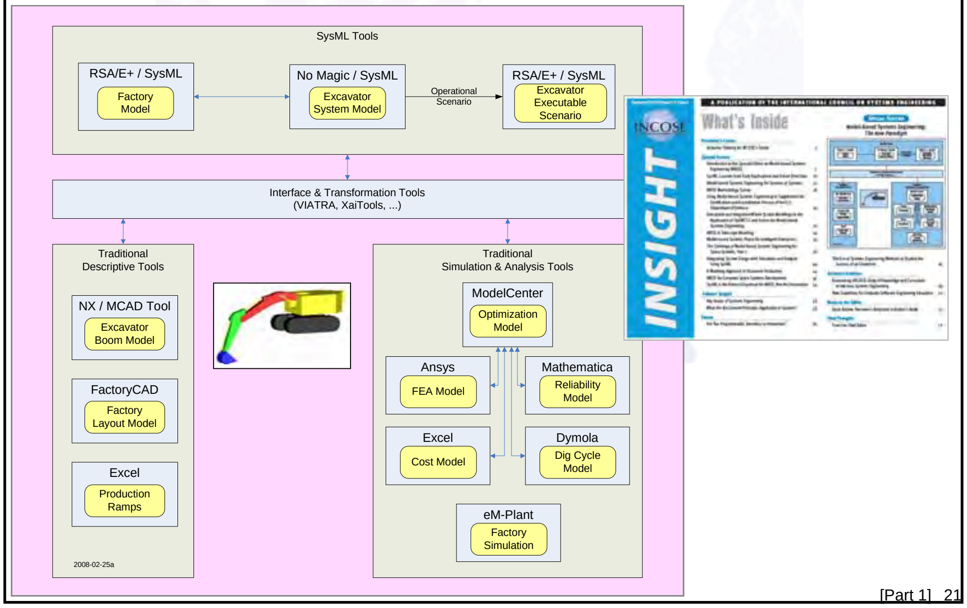
- Main team web page: *http://www.pslm.gatech.edu/projects/incose-mbse-msi/*
- These publications: *http://eislab.gatech.edu/pubs/seminars-etc/2009-04-ndia-ms/*

Contact

Russell.Peak@gatech.edu, Georgia Institute of Technology, Atlanta, www.msl.gatech.edu

Excavator Modeling & Simulation Testbed

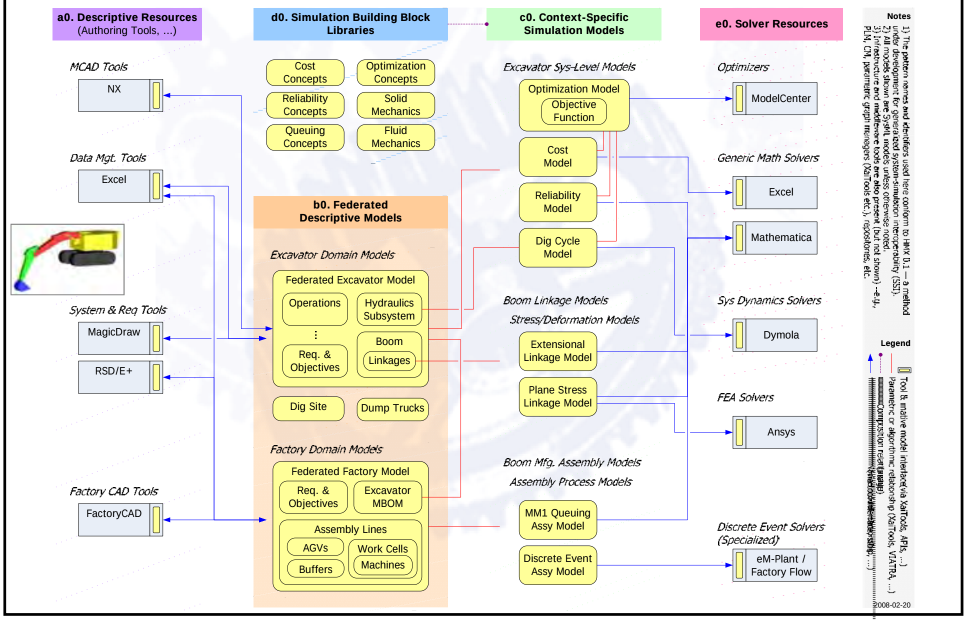
Tool Categories View



[Part 1] 21

Excavator Modeling & Simulation Testbed

Interoperability Patterns View (MSI Panorama per MIM patterns)



Notes

1) The pattern names and identifiers used here conform to HMX 0.1 — a method under development for generalized system-simulation interoperability (SSI).
2) All models shown are SysML models unless otherwise noted.
3) Infrastructure and middleware tools are also present (but not shown) --e.g., PLM, CM, parametric graph managers (XaiTools etc.), repositories, etc.

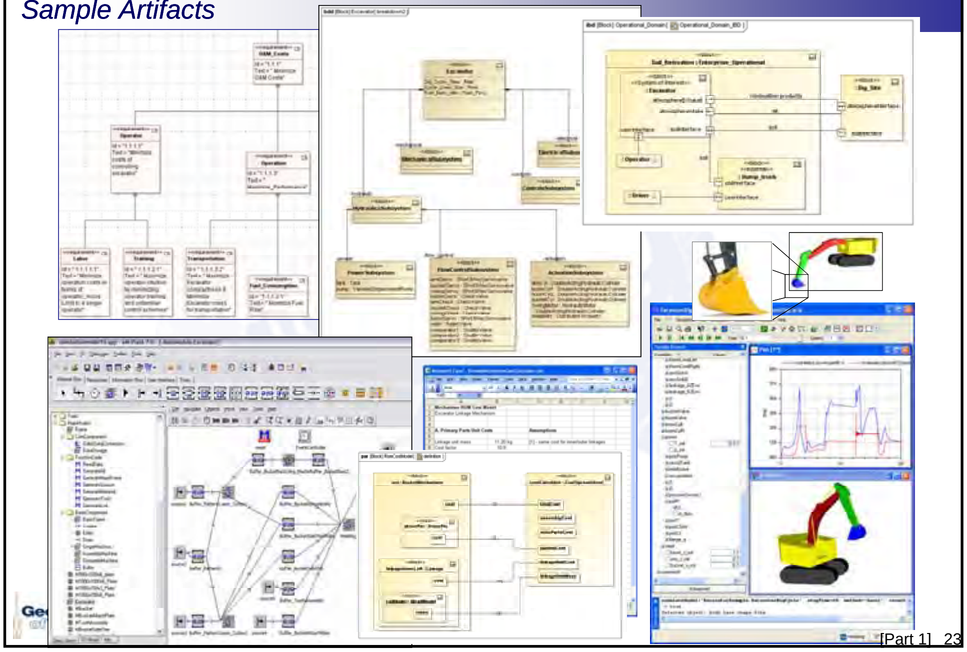
Legend

Tool & native model interface (via XaiTools, APIs, ...)
Parametric or algorithmic relationship (XaiTools, VIATRA, ...)
Composition relationship (usually)

2008-02-20

Excavator Modeling & Simulation Testbed

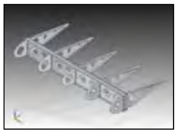
Sample Artifacts



[Part 1] 23

Broadly Applicable Technology

Examples of Executable SysML Parametrics



- Road scanning system using unmanned aerial vehicle (UAVs)
- UAV-based missile interceptor system trade study
- Space systems (tutorials): orbit planning; mass/cost roll-ups
- Space systems (studies/pilots): FireSat (INCOSE SSWG), ...
- Space systems (actuals): science merit function, ...
- Environmentally-conscious energy systems / smart grid
- Manufacturing “green-ness” / sustainability assessments
- Electronics recycling network
- Regional water management systems (e.g. South Florida)

…

- Mechanical part design and analysis (FEA)

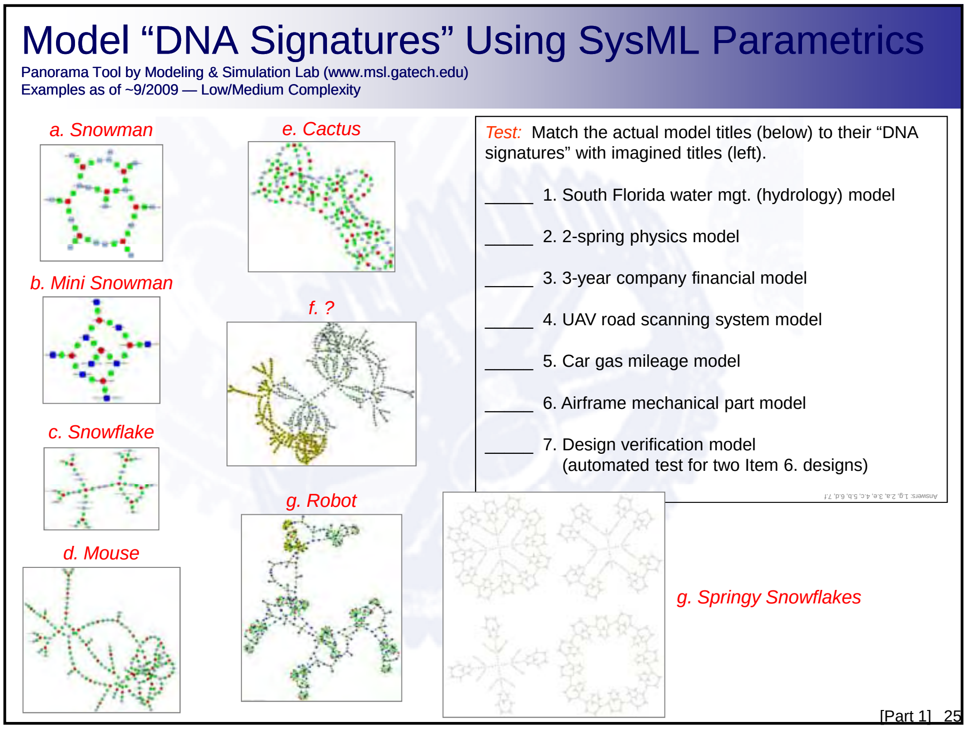
…

Next-Generation Spreadsheet Technology++

(object-oriented, multi-dimensional, ...)

- Wind turbine supply chain management
- Insurance claims processing and website capacity model
- Financial model for small businesses
- Banking service levels model

…



Model “DNA Signatures” Using SysML Parametrics

Panorama Tool by Modeling & Simulation Lab (www.msl.gatech.edu)
Examples as of ~9/2009 — Low/Medium Complexity

a. Snowman

b. Mini Snowman

c. Snowflake

d. Mouse

e. Cactus

f. ?

g. Robot

g. Springy Snowflakes

Test: Match the actual model titles (below) to their “DNA signatures” with imagined titles (left).

_____ 1. South Florida water mgt. (hydrology) model

_____ 2. 2-spring physics model

_____ 3. 3-year company financial model

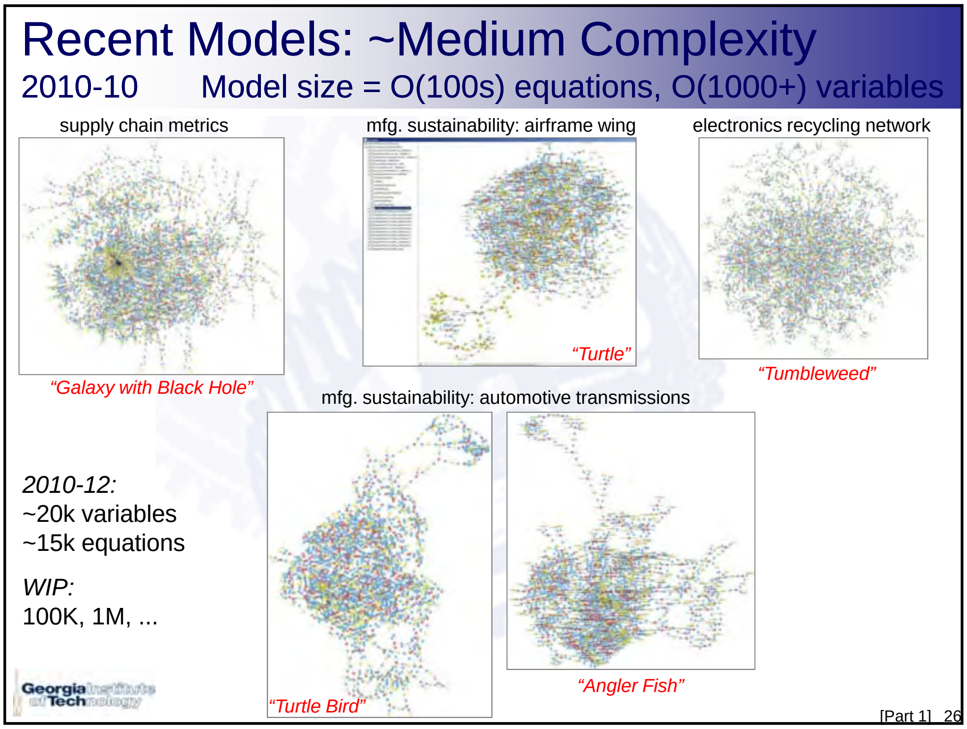
_____ 4. UAV road scanning system model

_____ 5. Car gas mileage model

_____ 6. Airframe mechanical part model

_____ 7. Design verification model
(automated test for two Item 6. designs)

Answers: 1.g, 2.a, 3.e, 4.c, 5.b, 6.d, 7.f



Recent Models: ~Medium Complexity

2010-10 Model size = O(100s) equations, O(1000+) variables

supply chain metrics

“Galaxy with Black Hole”

mfg. sustainability: airframe wing

“Turtle”

electronics recycling network

“Tumbleweed”

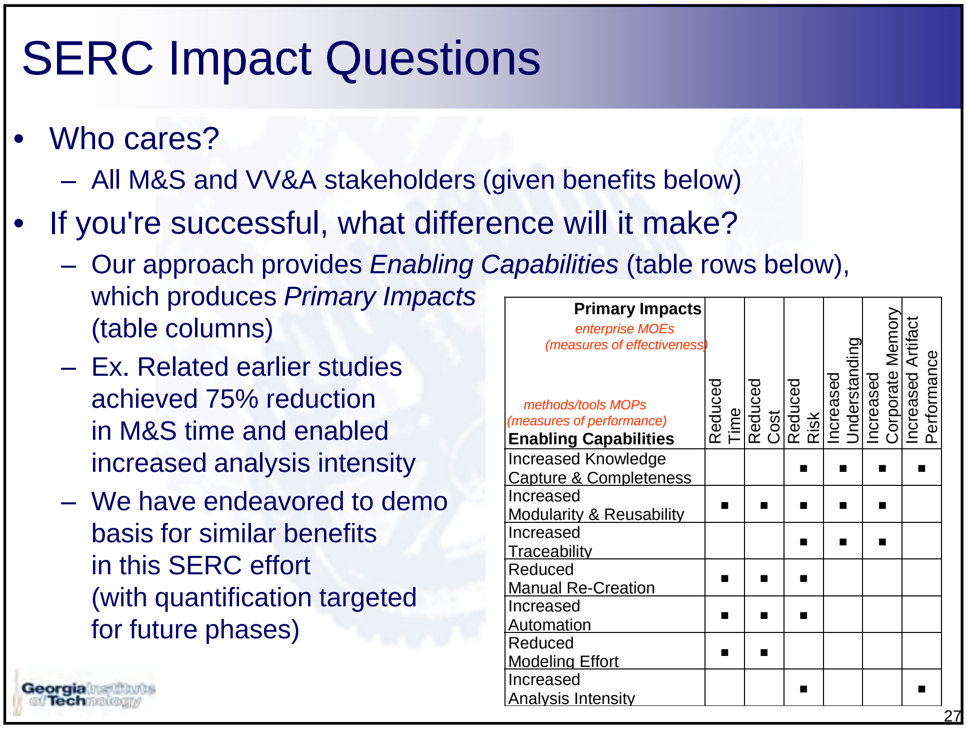
mfg. sustainability: automotive transmissions

“Turtle Bird”

“Angler Fish”

2010-12:
~20k variables
~15k equations

WIP:
100K, 1M, ...



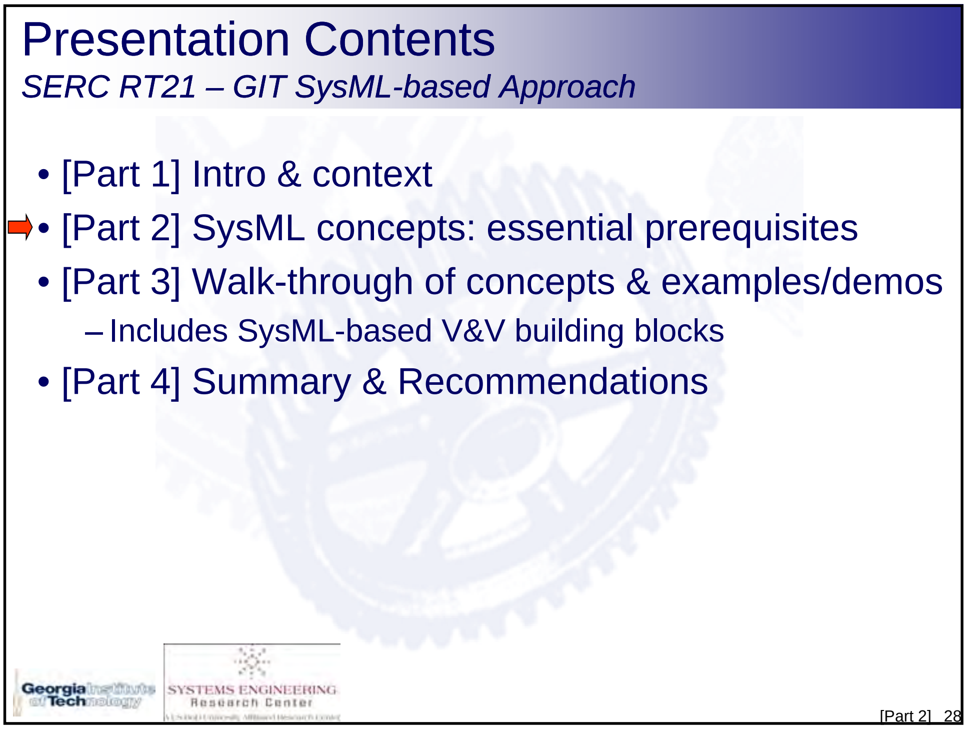
SERC Impact Questions

- Who cares?
  - All M&S and VV&A stakeholders (given benefits below)
- If you're successful, what difference will it make?
  - Our approach provides *Enabling Capabilities* (table rows below), which produces *Primary Impacts* (table columns)
  - Ex. Related earlier studies achieved 75% reduction in M&S time and enabled increased analysis intensity
  - We have endeavored to demo basis for similar benefits in this SERC effort (with quantification targeted for future phases)

Primary Impacts *enterprise MOEs* *(measures of effectiveness)* *methods/tools MOPs* *(measures of performance)* **Enabling Capabilities**	Reduced Time	Reduced Cost	Reduced Risk	Increased Understanding	Increased Corporate Memory	Increased Artifact Performance
Increased Knowledge Capture & Completeness			■	■	■	■
Increased Modularity & Reusability	■	■	■	■	■	
Increased Traceability			■	■	■	
Reduced Manual Re-Creation	■	■	■			
Increased Automation	■	■	■			
Reduced Modeling Effort	■	■				
Increased Analysis Intensity			■			■

Georgia Institute of Technology

27



Presentation Contents

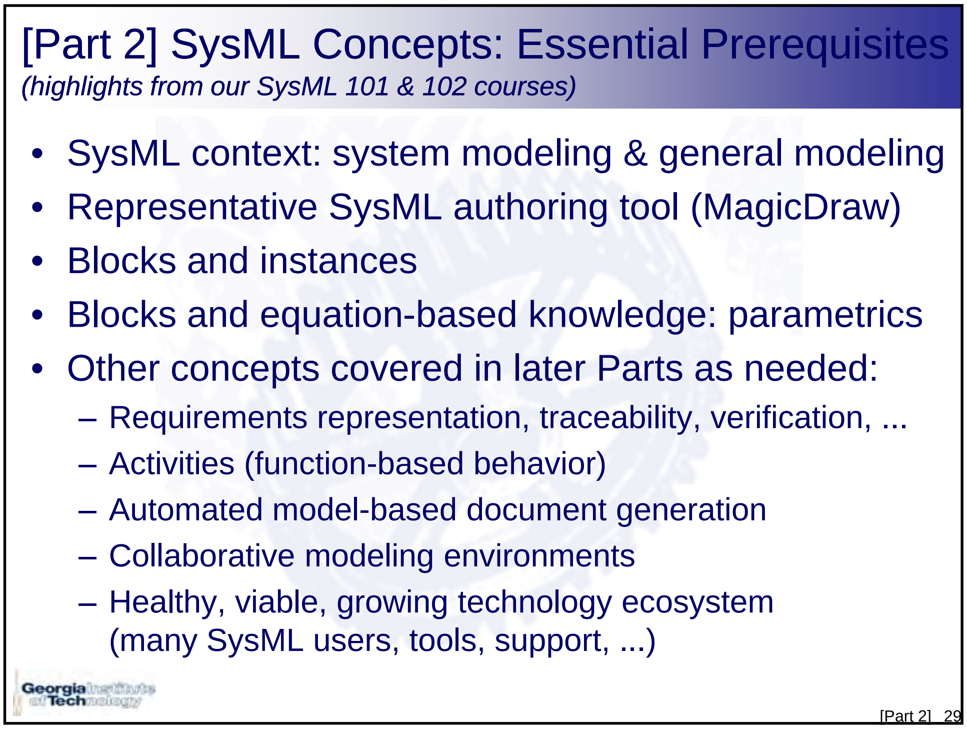
SERC RT21 – GIT SysML-based Approach

- [Part 1] Intro & context
- ➡ [Part 2] SysML concepts: essential prerequisites
- [Part 3] Walk-through of concepts & examples/demos
  - Includes SysML-based V&V building blocks
- [Part 4] Summary & Recommendations

Georgia Institute of Technology

SYSTEMS ENGINEERING Research Center

[Part 2] 28



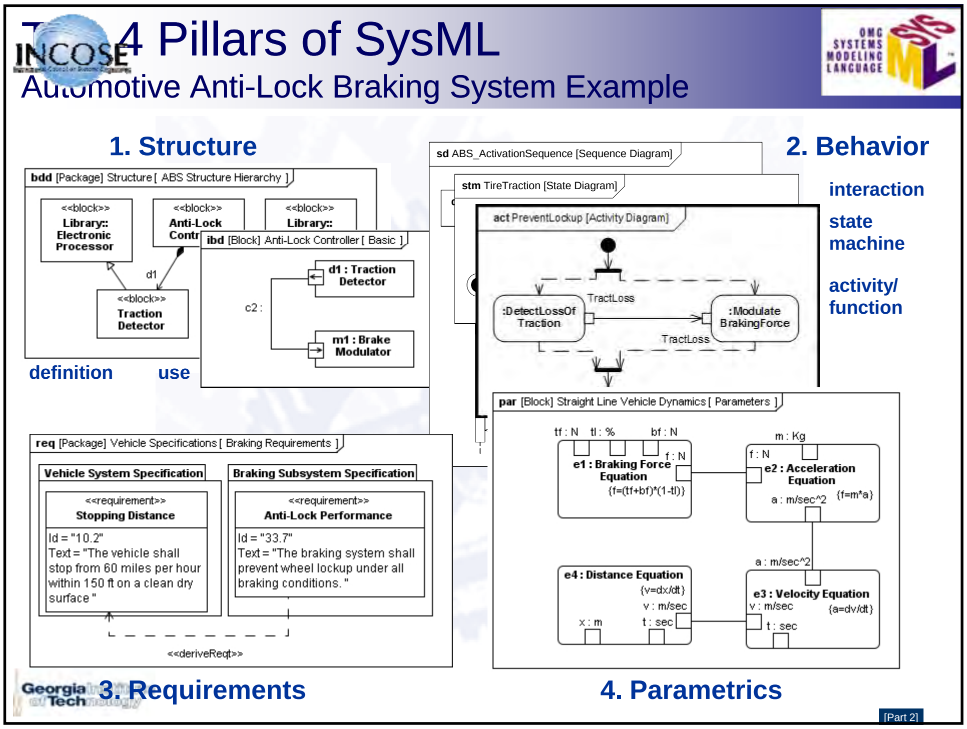
[Part 2] SysML Concepts: Essential Prerequisites

(highlights from our SysML 101 & 102 courses)

- SysML context: system modeling & general modeling
- Representative SysML authoring tool (MagicDraw)
- Blocks and instances
- Blocks and equation-based knowledge: parametrics
- Other concepts covered in later Parts as needed:
  - Requirements representation, traceability, verification, ...
  - Activities (function-based behavior)
  - Automated model-based document generation
  - Collaborative modeling environments
  - Healthy, viable, growing technology ecosystem (many SysML users, tools, support, ...)

Georgia Institute of Technology

[Part 2] 29



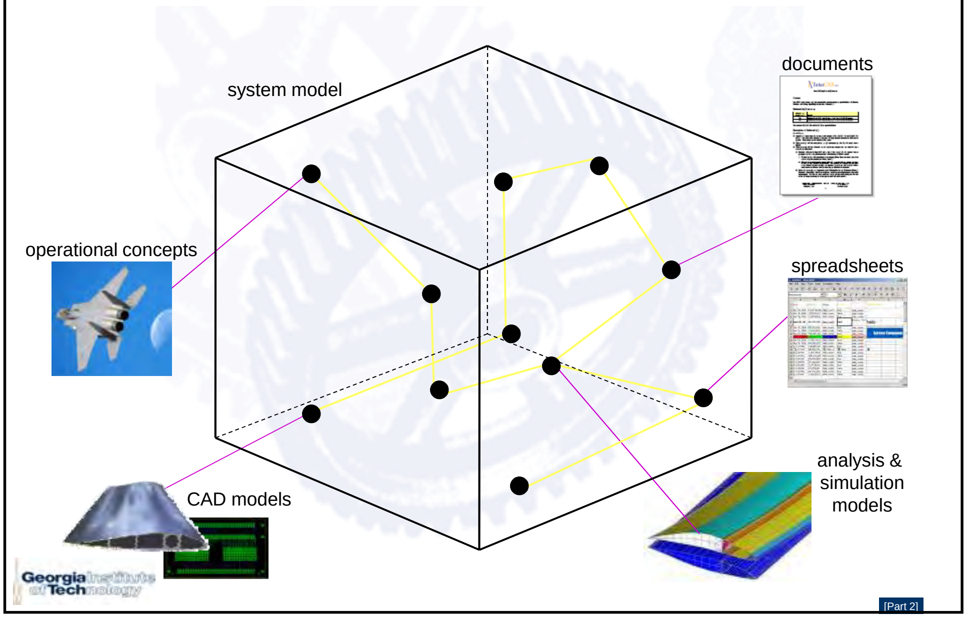
The 4 Pillars of SysML

Automotive Anti-Lock Braking System Example

INCOSE
OMG SYSTEMS MODELING LANGUAGE
1. Structure
bdd [Package] Structure [ABS Structure Hierarchy]
<<block>> Library:: Electronic Processor
<<block>> Anti-Lock Contr
<<block>> Library::
d1
<<block>> Traction Detector
ibd [Block] Anti-Lock Controller [Basic]
d1 : Traction Detector
c2 :
m1 : Brake Modulator
definition
use
2. Behavior
sd ABS_ActivationSequence [Sequence Diagram]
stm TireTraction [State Diagram]
act PreventLockup [Activity Diagram]
:DetectLossOf Traction
TractLoss
:Modulate BrakingForce
TractLoss
interaction
state machine
activity/ function
3. Requirements
req [Package] Vehicle Specifications [Braking Requirements]
Vehicle System Specification
Braking Subsystem Specification
<<requirement>> Stopping Distance
Id = "10.2"
Text = "The vehicle shall stop from 60 miles per hour within 150 ft on a clean dry surface "
<<requirement>> Anti-Lock Performance
Id = "33.7"
Text = "The braking system shall prevent wheel lockup under all braking conditions. "
<<deriveReqt>>
4. Parametrics
par [Block] Straight Line Vehicle Dynamics [Parameters]
tf : N
tl : %
bf : N
e1 : Braking Force Equation
f : N
{f=(tf+bf)*(1-tl)}
m : Kg
f : N
e2 : Acceleration Equation
a : m/sec^2
{f=m*a}
a : m/sec^2
e4 : Distance Equation
{v=dx/dt}
v : m/sec
x : m
t : sec
e3 : Velocity Equation
v : m/sec
{a=dv/dt}
t : sec
Georgia Institute of Technology
[Part 2]

Representing System Models

Without SysML: Ad-Hoc, Disconnected, Inconsistent, Implicit



system model

documents

operational concepts

spreadsheets

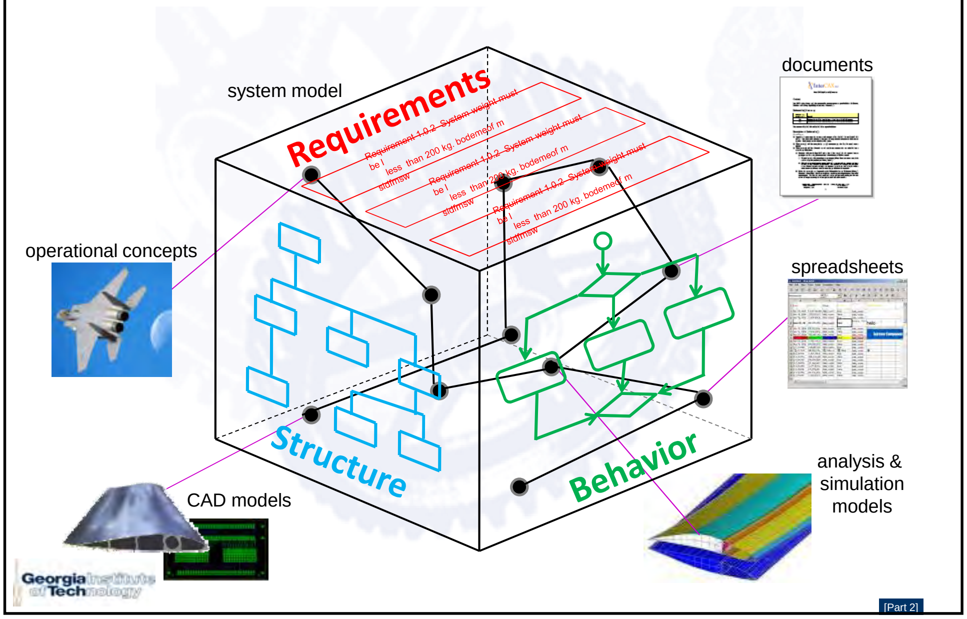
analysis & simulation models

CAD models

[Part 2]

Representing System Models

With SysML: Unified, Connected, Consistent, Explicit



system model

documents

operational concepts

spreadsheets

analysis & simulation models

CAD models

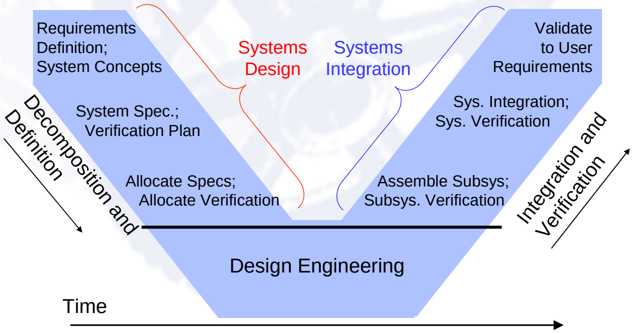
Requirements
Requirement 1.0.2 System weight must be l less than 200 kg. bodemeof m sldfmsw
Requirement 1.0.2 System weight must be l less than 200 kg. bodemeof m sldfmsw
Requirement 1.0.2 System weight must be l less than 200 kg. bodemeof m sldfmsw
Structure
Behavior

[Part 2]

What You Can Do with a SysML Model ...

System Modeling (and General Object Modeling)

- Describe requirements, system structure, & allocations
- Generate and/or link to simulations & verify requirements
- Visualize your models; Support system trade studies
- Link to domain models & analyses: S/W, M/ECAD, ...
- I.e., do the Vee and more ... (e.g., support system operation)


Requirements Definition; System Concepts
Systems Design
Systems Integration
Validate to User Requirements
System Spec.; Verification Plan
Sys. Integration; Sys. Verification
Decomposition and Definition
Allocate Specs; Allocate Verification
Assemble Subsys; Subsys. Verification
Integration and Verification
Design Engineering
Time

"Vee" model by Forsberg and Mooz, 1992

[Part 2]

Representative SysML Tools Used in RT21 Project

commercial tools: MagicDraw (base) + SysML plugin + ParaMagic plugin

prototype tools: Georgia Tech BuzzToys plugins: MyroMagic, Panorama, AeroMagic

Find out about our Unbeatable Performance Packages delivering Superior Results

Improve Enterprise Productivity with these Leading No Magic Products

MagicDraw

The multi award-winning UML business process, architecture, software and system modeling tool with teamwork support
See what's new in MagicDraw version 16.6
See what's new in upcoming MagicDraw 16.8 beta

Cameo™ Requirements+

Now with storyboarding the 4th generation business and software requirements management tool provides extreme flexibility for business users. Define business requirements that meet your process and business goals

MagicDraw SysML Plugin

SysML plugin supports all SysML diagrams, including Requirements, Block Definition, Internal Blocks, Parametric and others
See what's new in SysML plugin version 16.6
See what's new in upcoming SysML plugin 16.8 beta

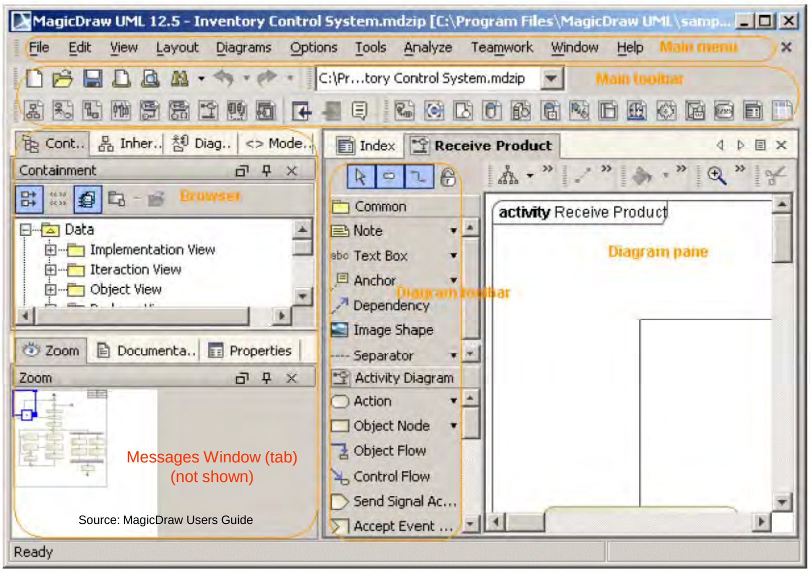
ParaMagic Plugin

ParaMagic plugin offers a dramatic expansion of the power of SysML parametric simulation, allowing users to integrate Microsoft Excel®, MATLAB® /Simulink® and Mathematica® into their MagicDraw SysML models

[Part 2]

MagicDraw Tool Fundamentals

User Interface Highlights



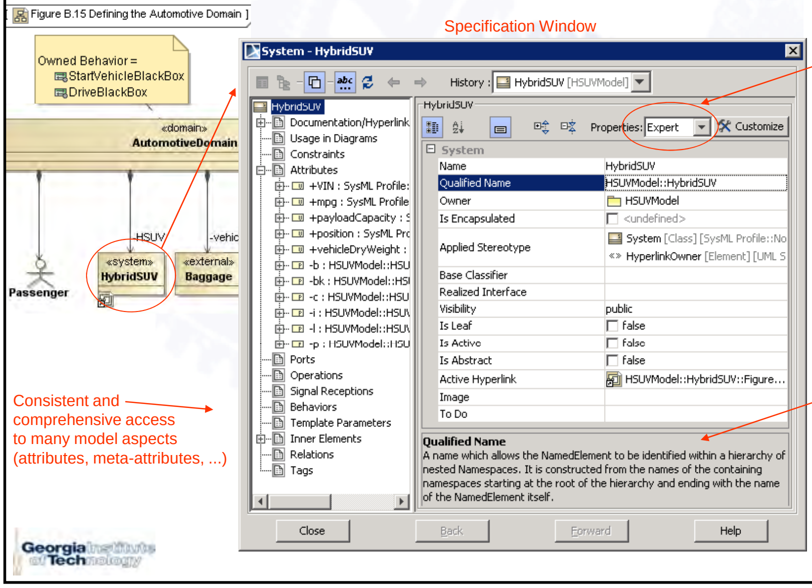
Tabs

Messages Window (tab) (not shown)

Source: MagicDraw Users Guide

Beyond Pretty Pictures: Rich Modeling Attributes (Metadata) in each SysML Block

Specification Window



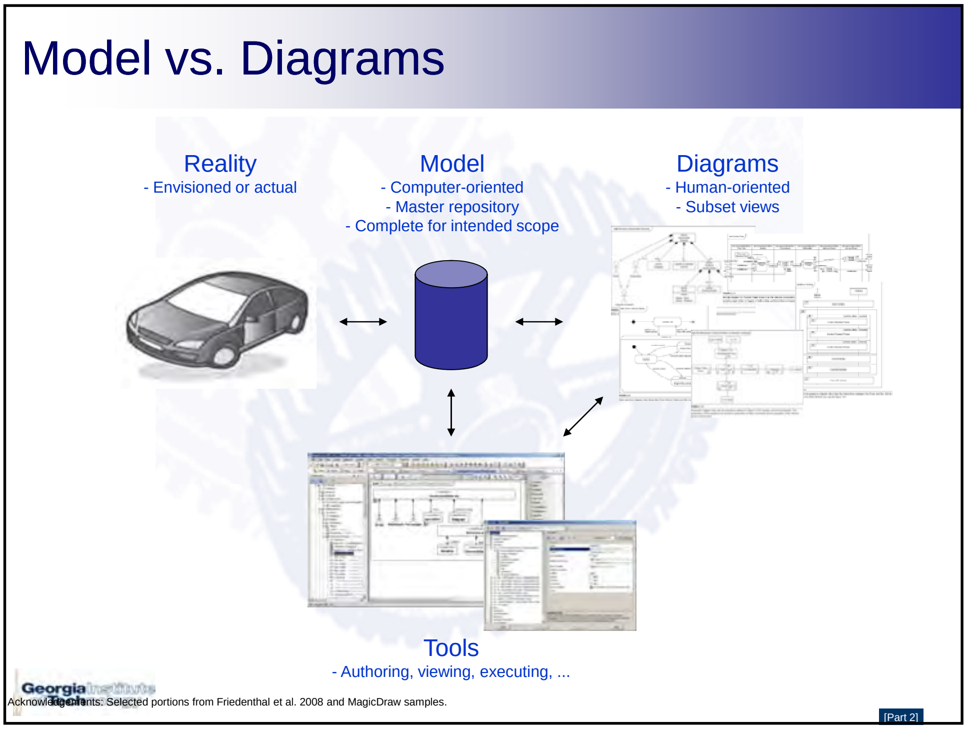
Name	HybridSUV
Qualified Name	HSUVModel::HybridSUV
Owner	HSUVModel
Is Encapsulated	□ <undefined>
Applied Stereotype	System [Class] [SysML Profile::No HyperlinkOwner [Element] [UML S
Base Classifier	
Realized Interface	
Visibility	public
Is Leaf	□ false
Is Active	□ false
Is Abstract	□ false
Active Hyperlink	HSUVModel::HybridSUV::Figure...
Image	
To Do	

Qualified Name
A name which allows the NamedElement to be identified within a hierarchy of nested Namespaces. It is constructed from the names of the containing namespaces starting at the root of the hierarchy and ending with the name of the NamedElement itself.

Filters to add/remove detail

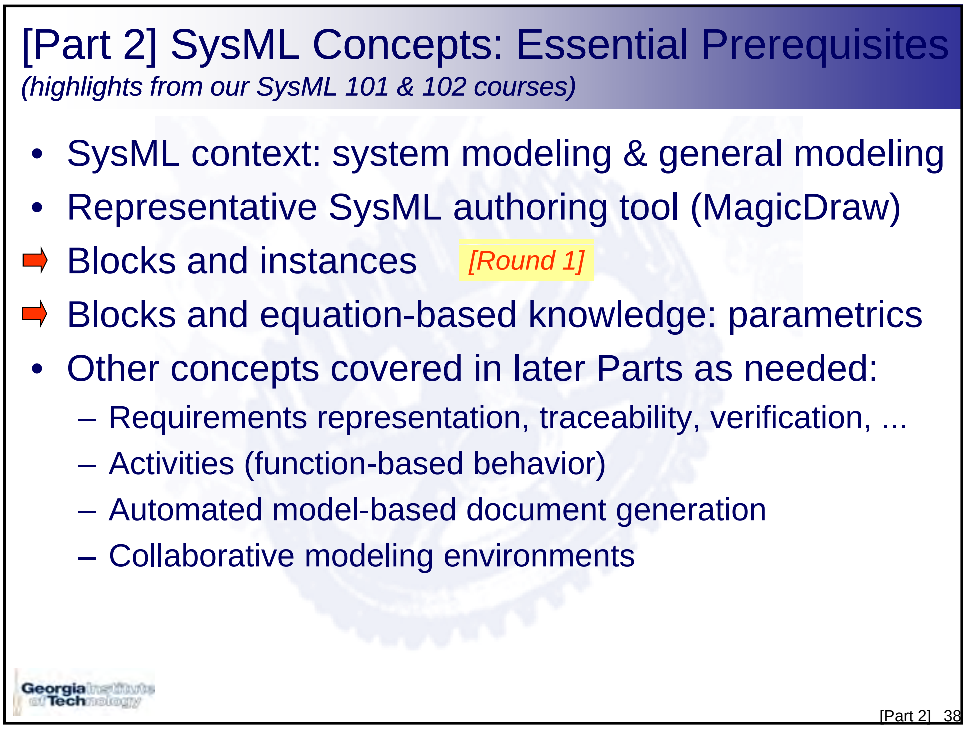
Built-in documentation on each model attribute

Consistent and comprehensive access to many model aspects (attributes, meta-attributes, ...)



Model vs. Diagrams

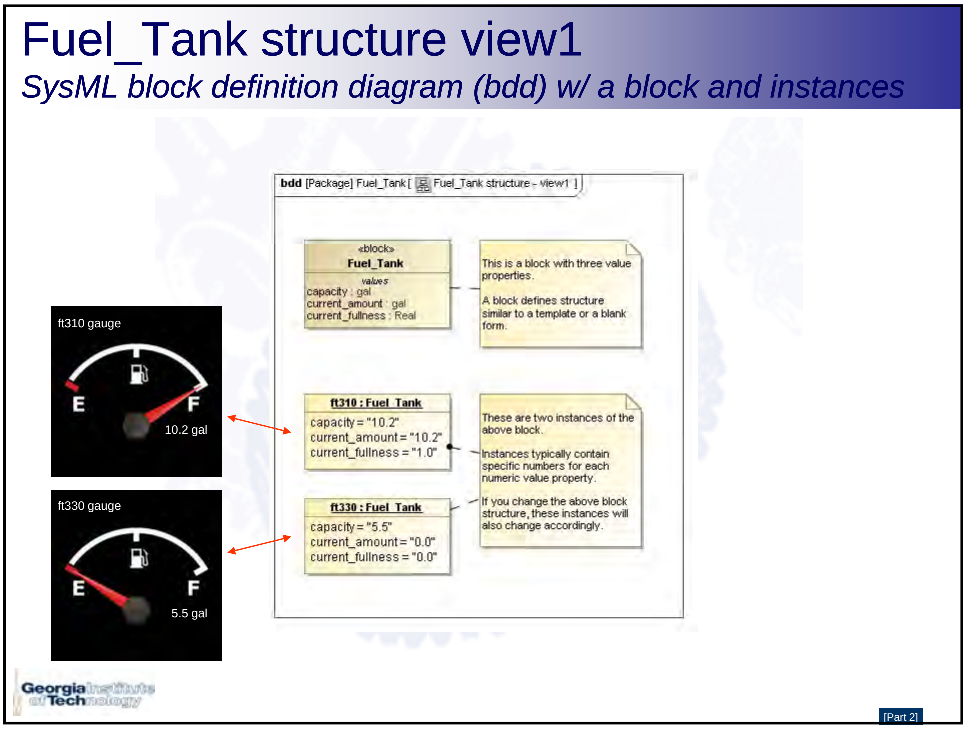
Acknowledgements: Selected portions from Friedenthal et al. 2008 and MagicDraw samples.



[Part 2] SysML Concepts: Essential Prerequisites

(highlights from our SysML 101 & 102 courses)

- SysML context: system modeling & general modeling
- Representative SysML authoring tool (MagicDraw)
- ➡ Blocks and instances *[Round 1]*
- ➡ Blocks and equation-based knowledge: parametrics
- Other concepts covered in later Parts as needed:
  - Requirements representation, traceability, verification, ...
  - Activities (function-based behavior)
  - Automated model-based document generation
  - Collaborative modeling environments



Fuel_Tank structure view1

SysML block definition diagram (bdd) w/ a block and instances

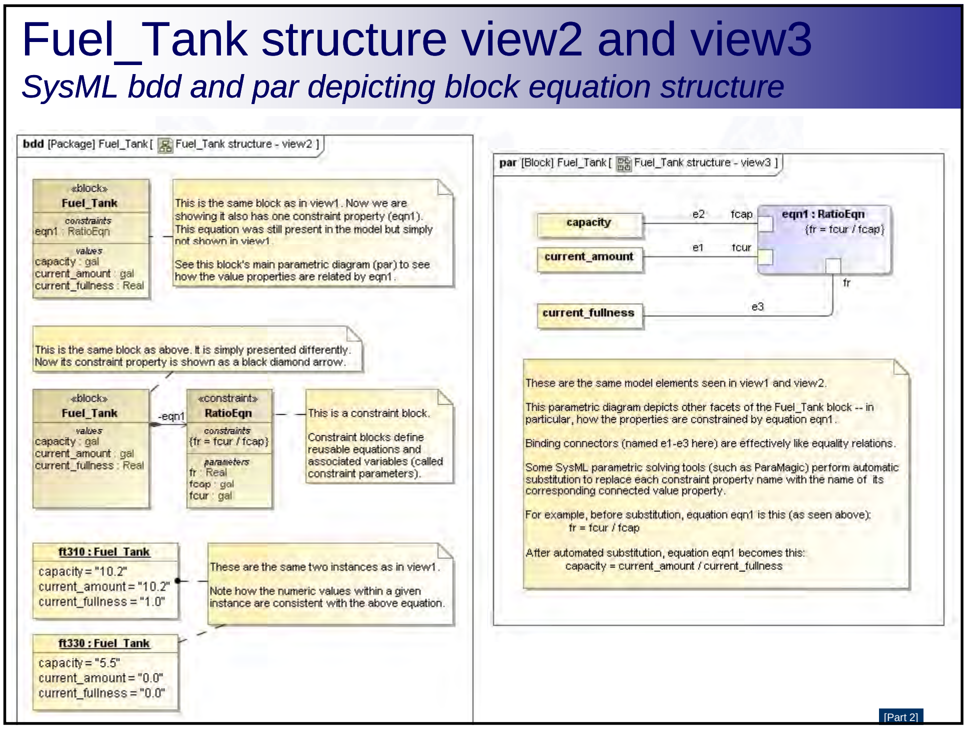
This is a block with three value properties.

A block defines structure similar to a template or a blank form.

These are two instances of the above block.

Instances typically contain specific numbers for each numeric value property.

If you change the above block structure, these instances will also change accordingly.



Fuel_Tank structure view2 and view3

SysML bdd and par depicting block equation structure

This is the same block as in view1. Now we are showing it also has one constraint property (eqn1). This equation was still present in the model but simply not shown in view1.

See this block's main parametric diagram (par) to see how the value properties are related by eqn1.

This is the same block as above. It is simply presented differently. Now its constraint property is shown as a black diamond arrow.

This is a constraint block.

Constraint blocks define reusable equations and associated variables (called constraint parameters).

These are the same two instances as in view1.

Note how the numeric values within a given instance are consistent with the above equation.

These are the same model elements seen in view1 and view2.

This parametric diagram depicts other facets of the Fuel_Tank block -- in particular, how the properties are constrained by equation eqn1.

Binding connectors (named e1-e3 here) are effectively like equality relations.

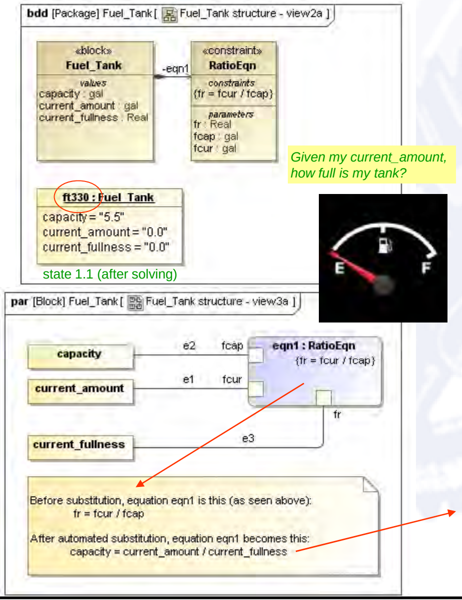
Some SysML parametric solving tools (such as ParaMagic) perform automatic substitution to replace each constraint property name with the name of its corresponding connected value property.

For example, before substitution, equation eqn1 is this (as seen above):
fr = fcur / fcap

After automated substitution, equation eqn1 becomes this:
capacity = current_amount / current_fullness

Fuel_Tank parametrics execution

ParaMagic interoperating w/ equation solvers such as Mathematica



bdd [Package] Fuel_Tank [Fuel_Tank structure - view2a]

«block»
Fuel_Tank
values
capacity : gal
current_amount : gal
current_fullness : Real

-eqn1

«constraint»
RatioEqn
constraints
{fr = fcur / fcap}
parameters
fr : Real
fcap : gal
fcur : gal

Given my current_amount, how full is my tank?

ft330 : Fuel_Tank
capacity = "5.5"
current_amount = "0.0"
current_fullness = "0.0"

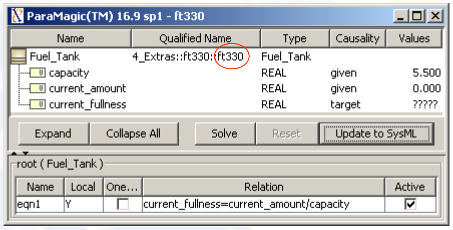
state 1.1 (after solving)

par [Block] Fuel_Tank [Fuel_Tank structure - view3a]

Before substitution, equation eqn1 is this (as seen above):
fr = fcur / fcap

After automated substitution, equation eqn1 becomes this:
capacity = current_amount / current_fullness

instance ft330 state 1.0 (before solving)



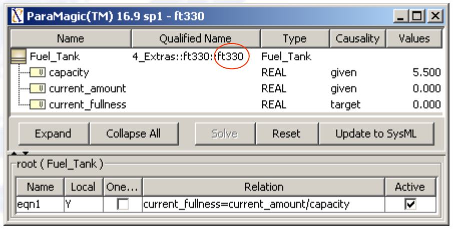
ParaMagic(TM) 16.9 sp1 - ft330

Name	Qualified Name	Type	Causality	Values
Fuel_Tank	4_Extras::ft330::ft330	Fuel_Tank		
capacity		REAL	given	5.500
current_amount		REAL	given	0.000
current_fullness		REAL	target	?????

root (Fuel_Tank)

Name	Local	One...	Relation	Active
eqn1	Y	☐	current_fullness=current_amount/capacity	☑

instance ft330 state 1.1 (after solving)



ParaMagic(TM) 16.9 sp1 - ft330

Name	Qualified Name	Type	Causality	Values
Fuel_Tank	4_Extras::ft330::ft330	Fuel_Tank		
capacity		REAL	given	5.500
current_amount		REAL	given	0.000
current_fullness		REAL	target	0.000

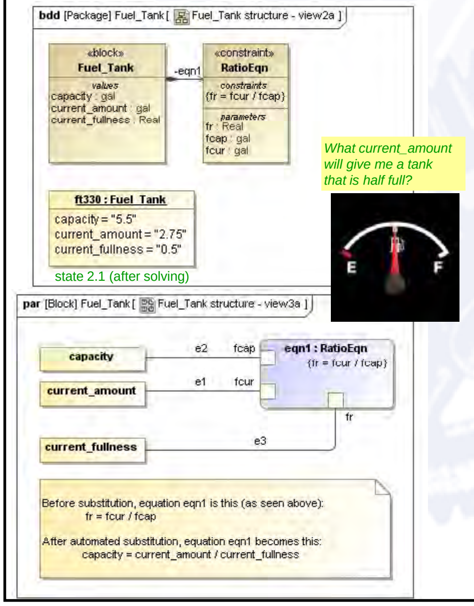
root (Fuel_Tank)

Name	Local	One...	Relation	Active
eqn1	Y	☐	current_fullness=current_amount/capacity	☑

[Part 2]

Fuel_Tank parametrics execution

Changing input/output direction (causality) in the same instance



bdd [Package] Fuel_Tank [Fuel_Tank structure - view2a]

«block»
Fuel_Tank
values
capacity : gal
current_amount : gal
current_fullness : Real

-eqn1

«constraint»
RatioEqn
constraints
{fr = fcur / fcap}
parameters
fr : Real
fcap : gal
fcur : gal

What current_amount will give me a tank that is half full?

ft330 : Fuel_Tank
capacity = "5.5"
current_amount = "2.75"
current_fullness = "0.5"

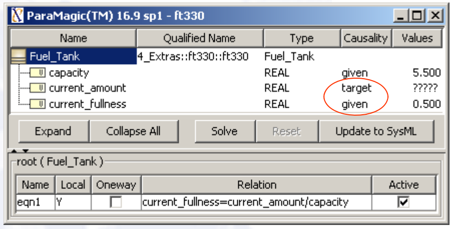
state 2.1 (after solving)

par [Block] Fuel_Tank [Fuel_Tank structure - view3a]

Before substitution, equation eqn1 is this (as seen above):
fr = fcur / fcap

After automated substitution, equation eqn1 becomes this:
capacity = current_amount / current_fullness

instance ft330 state 2.0 (after changing causalities, and before solving)



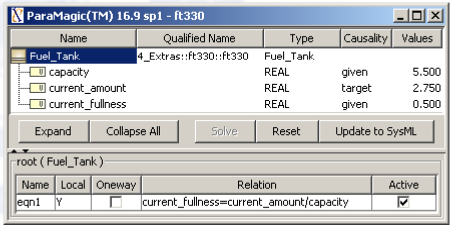
ParaMagic(TM) 16.9 sp1 - ft330

Name	Qualified Name	Type	Causality	Values
Fuel_Tank	4_Extras::ft330::ft330	Fuel_Tank		
capacity		REAL	given	5.500
current_amount		REAL	target	?????
current_fullness		REAL	given	0.500

root (Fuel_Tank)

Name	Local	Oneway	Relation	Active
eqn1	Y	☐	current_fullness=current_amount/capacity	☑

instance ft330 state 2.1 (after solving)



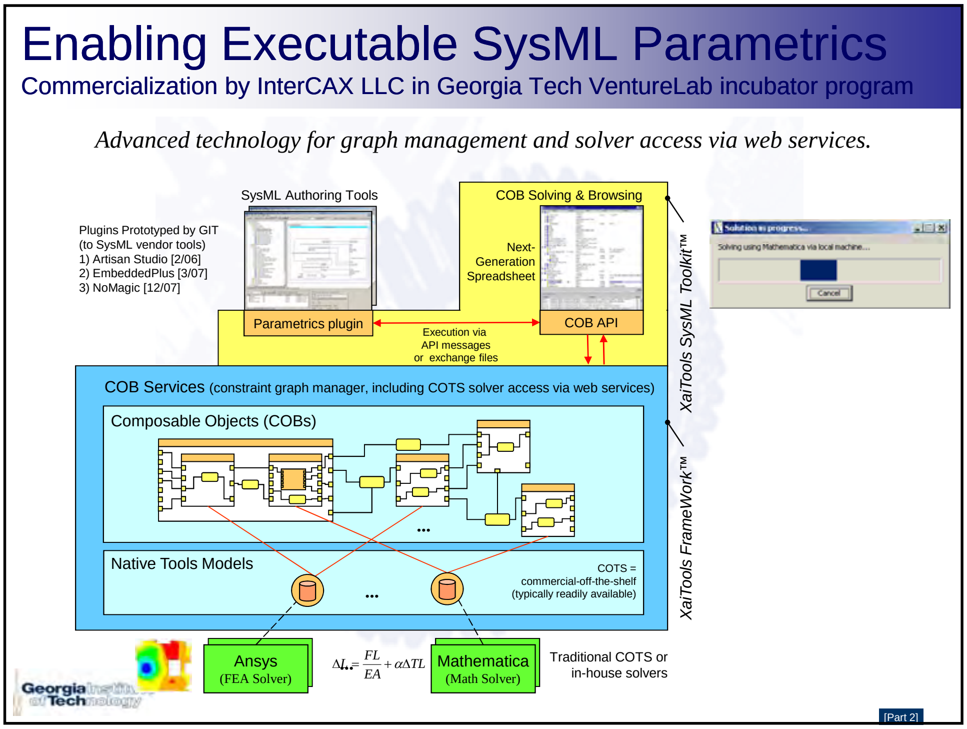
ParaMagic(TM) 16.9 sp1 - ft330

Name	Qualified Name	Type	Causality	Values
Fuel_Tank	4_Extras::ft330::ft330	Fuel_Tank		
capacity		REAL	given	5.500
current_amount		REAL	target	2.750
current_fullness		REAL	given	0.500

root (Fuel_Tank)

Name	Local	Oneway	Relation	Active
eqn1	Y	☐	current_fullness=current_amount/capacity	☑

[Part 2]

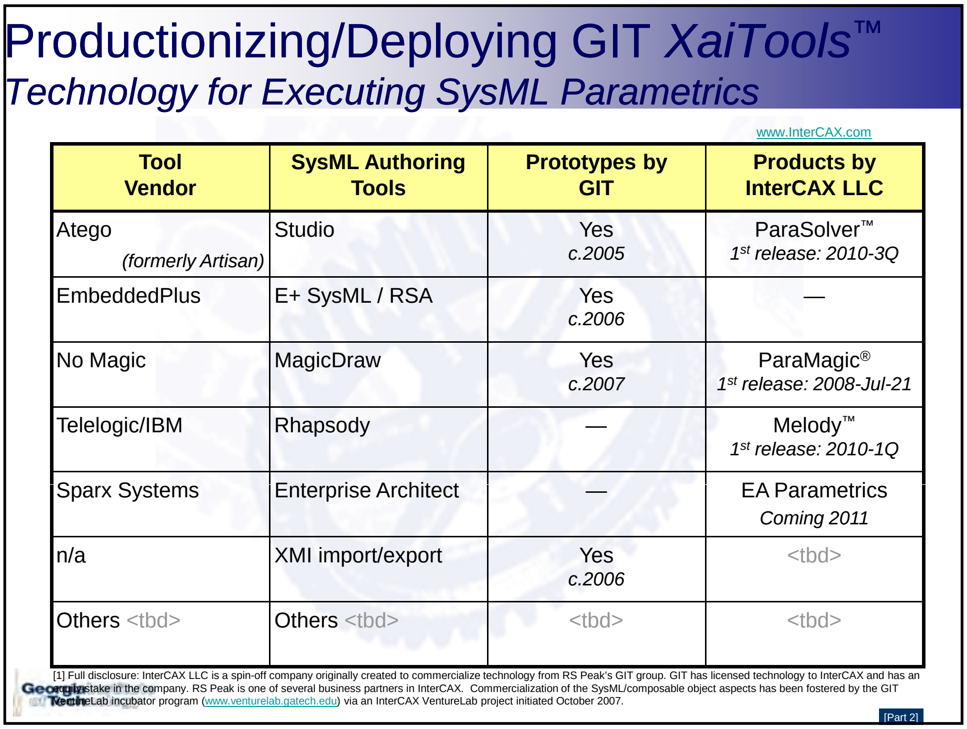


Enabling Executable SysML Parametrics

Commercialization by InterCAX LLC in Georgia Tech VentureLab incubator program

Advanced technology for graph management and solver access via web services.

SysML Authoring Tools
Plugins Prototyped by GIT
(to SysML vendor tools)
1) Artisan Studio [2/06]
2) EmbeddedPlus [3/07]
3) NoMagic [12/07]
COB Solving & Browsing
Next-Generation Spreadsheet
Solution in progress...
Solving using Mathematica via local machine...
Cancel
Parametrics plugin
Execution via API messages or exchange files
COB API
COB Services (constraint graph manager, including COTS solver access via web services)
Composable Objects (COBs)
...
Native Tools Models
...
COTS = commercial-off-the-shelf (typically readily available)
XaiTools SysML Toolkit™
XaiTools FrameWork™
Ansys (FEA Solver)
$\Delta L = \frac{FL}{EA} + \alpha\Delta TL$
Mathematica (Math Solver)
Traditional COTS or in-house solvers
Georgia Institute of Technology
[Part 2]



Productionizing/Deploying GIT *XaiTools*™

Technology for Executing SysML Parametrics

www.InterCAX.com

Tool Vendor	SysML Authoring Tools	Prototypes by GIT	Products by InterCAX LLC
Atego *(formerly Artisan)*	Studio	Yes *c.2005*	ParaSolver™ *1st release: 2010-3Q*
EmbeddedPlus	E+ SysML / RSA	Yes *c.2006*	—
No Magic	MagicDraw	Yes *c.2007*	ParaMagic® *1st release: 2008-Jul-21*
Telelogic/IBM	Rhapsody	—	Melody™ *1st release: 2010-1Q*
Sparx Systems	Enterprise Architect	—	EA Parametrics *Coming 2011*
n/a	XMI import/export	Yes *c.2006*	<tbd>
Others <tbd>	Others <tbd>	<tbd>	<tbd>

[1] Full disclosure: InterCAX LLC is a spin-off company originally created to commercialize technology from RS Peak's GIT group. GIT has licensed technology to InterCAX and has an equity stake in the company. RS Peak is one of several business partners in InterCAX. Commercialization of the SysML/composable object aspects has been fostered by the GIT VentureLab incubator program (www.venturelab.gatech.edu) via an InterCAX VentureLab project initiated October 2007.

Inte

Pro

www.InterCAX.com

InterCAX

Modeling & Simulation Technology

Products

SysML/UPDM Parametric Solvers

- Execute parametric models using
  - ParaMagic® for MagicDraw
  - Melody™ for Rational Rhapsody
  - ParaSolver™ for Artisan Studio
  - Solvea™ (beta) for Enterprise Architect
- Use Mathematica or OpenModelica (free) as a core solver
- Link to Excel spreadsheets and perform batch trade studies
- Link to existing MATLAB/Simulink and Mathematica functions

SysML/UPDM Integrations

- Math tools (Mathematica, MATLAB, Excel)
- M&S tools (Simulink, STK, OpenModelica)
- CAD tools (NX, STEP MCAD/AP203, ECAD/AP210)
- CAE tools (ANSYS, ABAQUS)
- Plus tailored interfaces (Teamcenter/PLM, in-house tools)

Services

- SysML training – onsite, offsite, and web-based
  www.intercax.com/sysml/training
- Consulting, coaching, and support for SysML & MBSE
- Custom application development

Application Areas

- Aerospace, automotive, and marine systems
- Defense and intelligence
- Supply chains and logistics
- Energy and sustainability
- Finance and risk management
- Urban planning and infrastructure

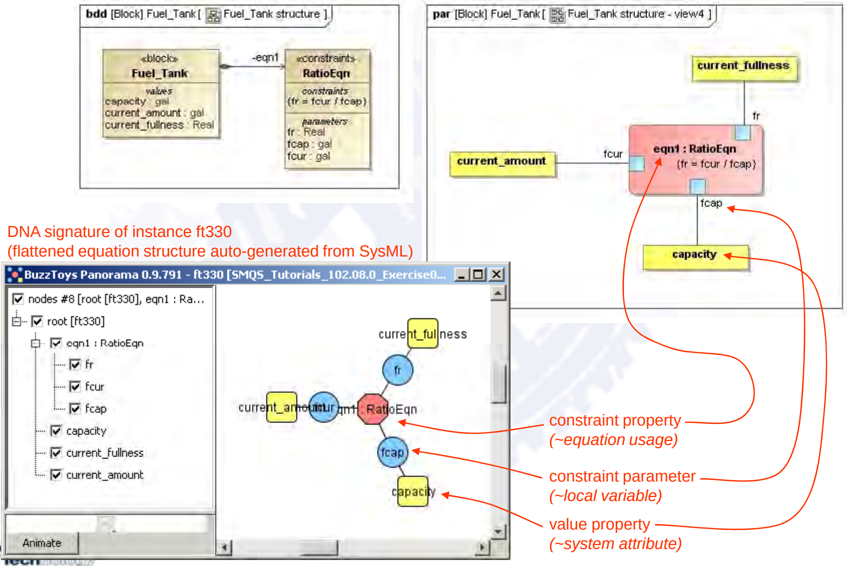
Contact Us

75 5th Street NW, Suite 213, Atlanta GA 30318, USA
info@intercax.com • +1-404-592-6897 • twitter.com/InterCAX

[Part 2]

Fuel_Tank “DNA signature”

Interacting with equation graph structure via Panorama tool



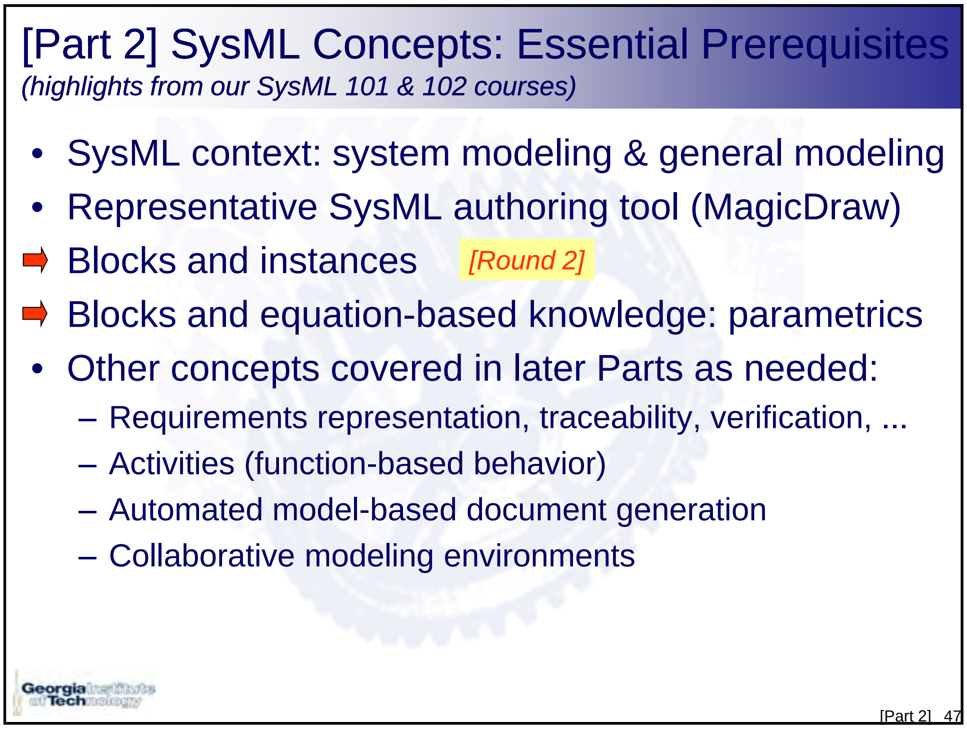
DNA signature of instance ft330
(flattened equation structure auto-generated from SysML)

constraint property
(~equation usage)

constraint parameter
(~local variable)

value property
(~system attribute)

[Part 2]



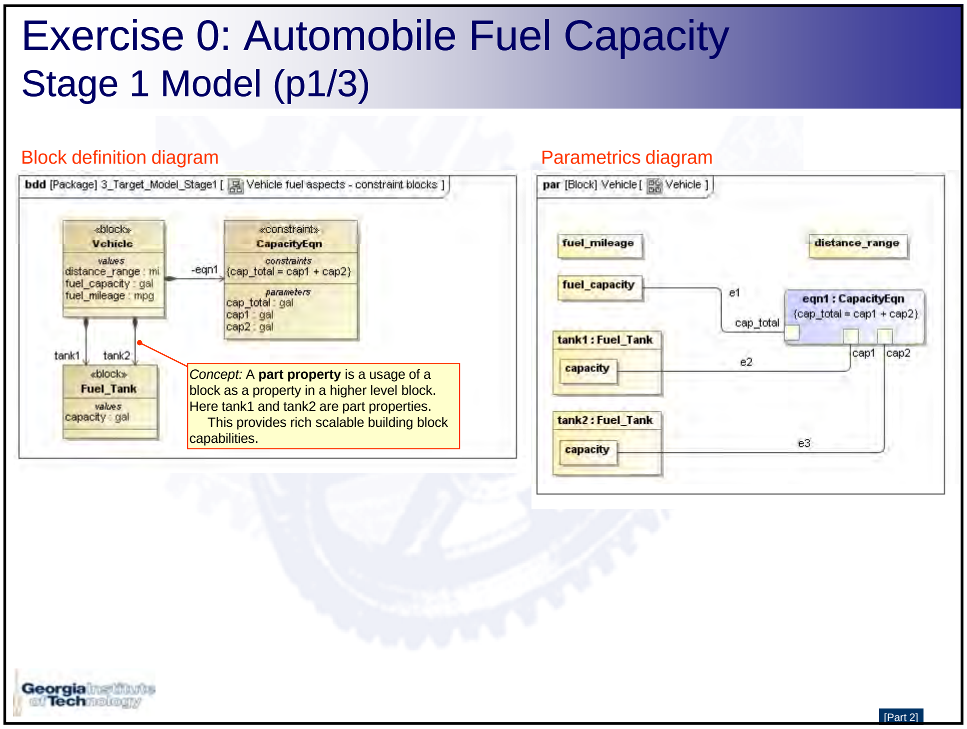
[Part 2] SysML Concepts: Essential Prerequisites

(highlights from our SysML 101 & 102 courses)

- SysML context: system modeling & general modeling
- Representative SysML authoring tool (MagicDraw)
- ➡ Blocks and instances *[Round 2]*
- ➡ Blocks and equation-based knowledge: parametrics
- Other concepts covered in later Parts as needed:
  - Requirements representation, traceability, verification, ...
  - Activities (function-based behavior)
  - Automated model-based document generation
  - Collaborative modeling environments

Georgia Institute of Technology

[Part 2] 47



Exercise 0: Automobile Fuel Capacity
Stage 1 Model (p1/3)

Block definition diagram

Concept: A **part property** is a usage of a block as a property in a higher level block. Here tank1 and tank2 are part properties.
This provides rich scalable building block capabilities.

Parametrics diagram

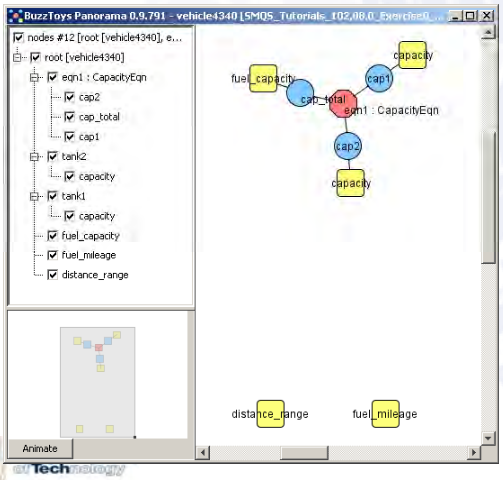
Georgia Institute of Technology

[Part 2]

Exercise 0: Automobile Fuel Capacity
Stage 1 Model (p2/3)

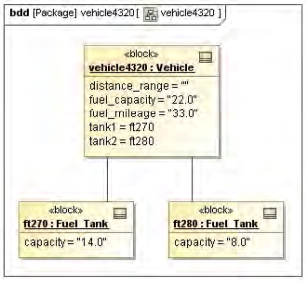
Model DNA Signature

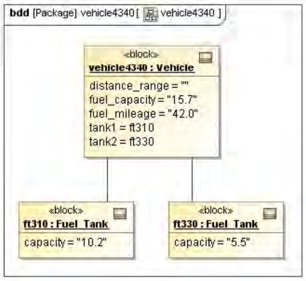
Example Instances (after solving)



BuzzToys Panorama 0.9.791 - vehicle4340 [SMQS_Tutorials_102,08.0_Exercise0_...

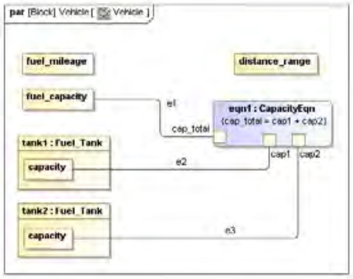
- ☑ nodes #12 [root [vehicle4340], e...
  - ☑ root [vehicle4340]
    - ☑ eqn1 : CapacityEqn
      - ☑ cap2
      - ☑ cap_total
      - ☑ cap1
    - ☑ tank2
      - ☑ capacity
    - ☑ tank1
      - ☑ capacity
    - ☑ fuel_capacity
    - ☑ fuel_mileage
    - ☑ distance_range

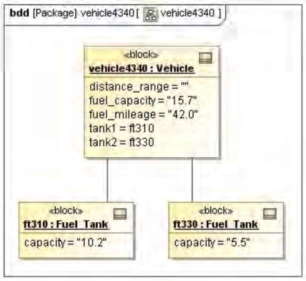


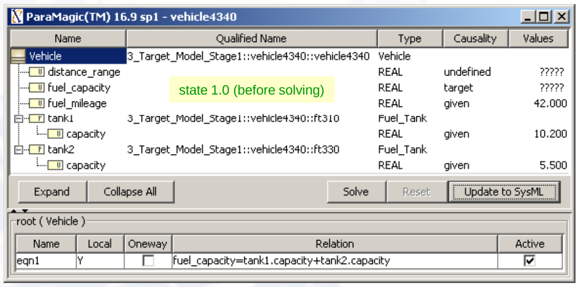


[Part 2]

Exercise 0: Automobile Fuel Capacity
Stage 1 Model (p3/3)







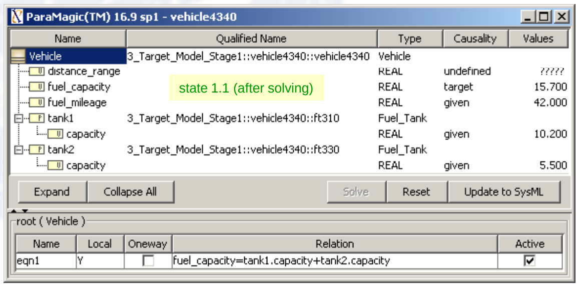
ParaMagic(TM) 16.9 sp1 - vehicle4340

state 1.0 (before solving)

Name	Qualified Name	Type	Causality	Values
Vehicle	3_Target_Model_Stage1::vehicle4340::vehicle4340	Vehicle		
distance_range		REAL	undefined	?????
fuel_capacity		REAL	target	?????
fuel_mileage		REAL	given	42.000
tank1	3_Target_Model_Stage1::vehicle4340::ft310	Fuel_Tank		
capacity		REAL	given	10.200
tank2	3_Target_Model_Stage1::vehicle4340::ft330	Fuel_Tank		
capacity		REAL	given	5.500

root (Vehicle)

Name	Local	Oneway	Relation	Active
eqn1	Y	☐	fuel_capacity=tank1.capacity+tank2.capacity	☑



ParaMagic(TM) 16.9 sp1 - vehicle4340

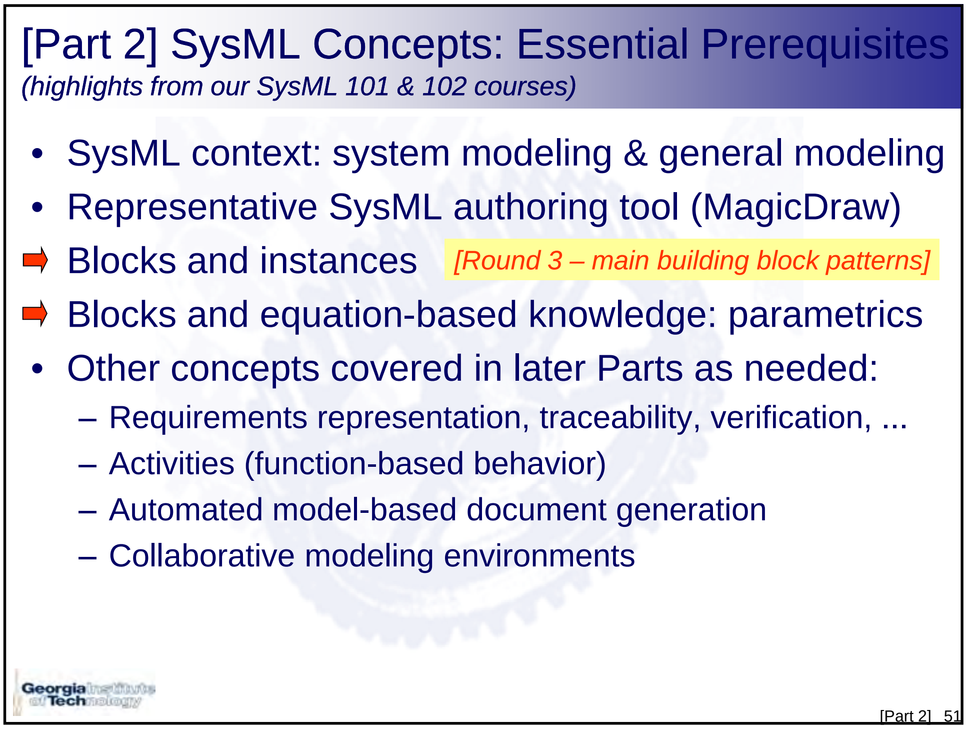
state 1.1 (after solving)

Name	Qualified Name	Type	Causality	Values
Vehicle	3_Target_Model_Stage1::vehicle4340::vehicle4340	Vehicle		
distance_range		REAL	undefined	?????
fuel_capacity		REAL	target	15.700
fuel_mileage		REAL	given	42.000
tank1	3_Target_Model_Stage1::vehicle4340::ft310	Fuel_Tank		
capacity		REAL	given	10.200
tank2	3_Target_Model_Stage1::vehicle4340::ft330	Fuel_Tank		
capacity		REAL	given	5.500

root (Vehicle)

Name	Local	Oneway	Relation	Active
eqn1	Y	☐	fuel_capacity=tank1.capacity+tank2.capacity	☑

[Part 2]

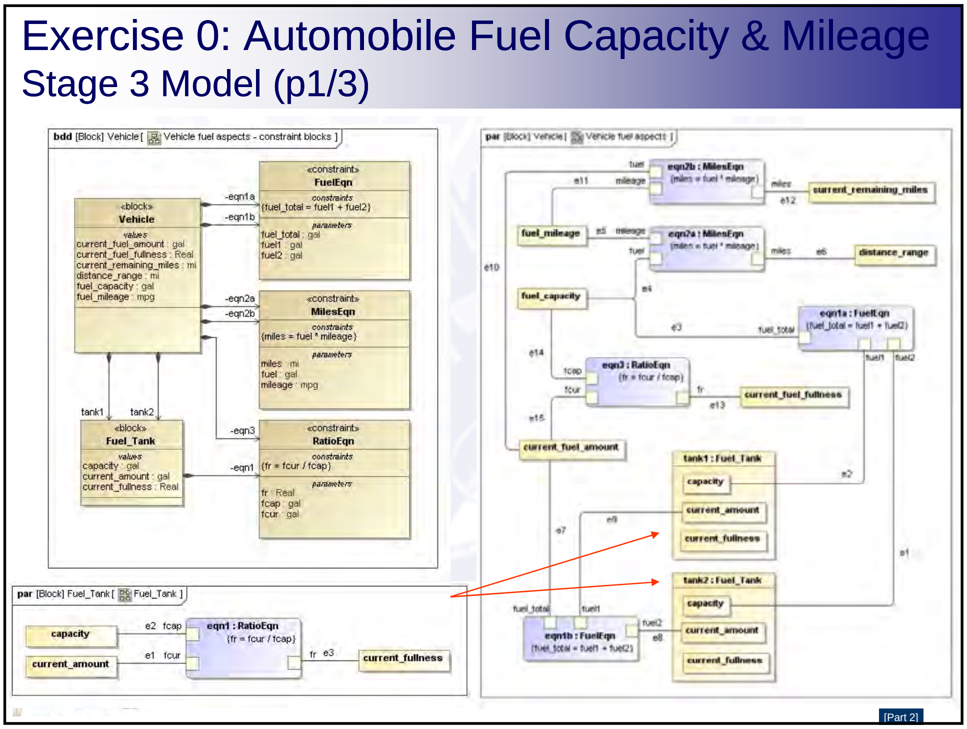


[Part 2] SysML Concepts: Essential Prerequisites

(highlights from our SysML 101 & 102 courses)

- SysML context: system modeling & general modeling
- Representative SysML authoring tool (MagicDraw)
- ➡ Blocks and instances *[Round 3 – main building block patterns]*
- ➡ Blocks and equation-based knowledge: parametrics
- Other concepts covered in later Parts as needed:
  - Requirements representation, traceability, verification, ...
  - Activities (function-based behavior)
  - Automated model-based document generation
  - Collaborative modeling environments

[Part 2] 51

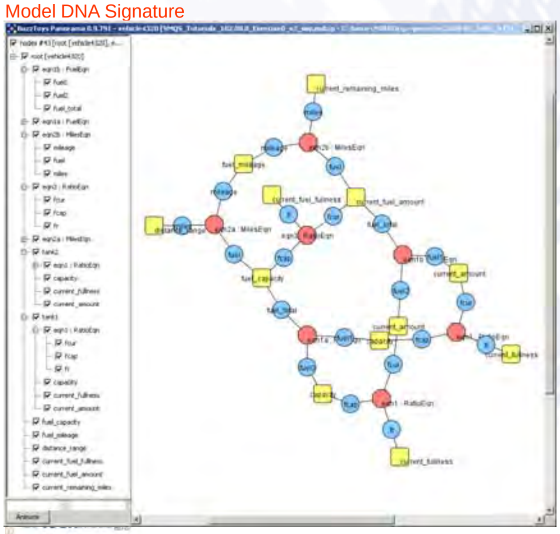


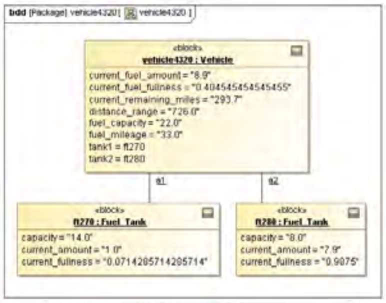
Exercise 0: Automobile Fuel Capacity & Mileage
Stage 3 Model (p1/3)

Exercise 0: Automobile Fuel Capacity & Mileage
Stage 3 Model (p2/3)

Example Instances
(after solving)

Model DNA Signature



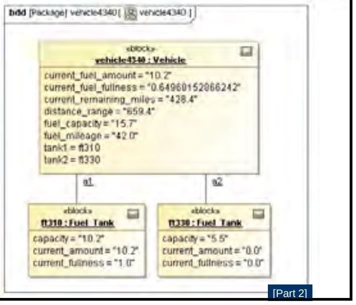


bdd [Package] vehicle4320 [vehicle4320]

«block»
vehicle4320 : Vehicle
current_fuel_amount = "8.9"
current_fuel_fullness = "0.40454545454545455"
current_remaining_miles = "293.7"
distance_range = "726.0"
fuel_capacity = "22.0"
fuel_mileage = "33.0"
tank1 = ft270
tank2 = ft280

«block»
ft270 : Fuel_Tank
capacity = "14.0"
current_amount = "1.0"
current_fullness = "0.07142857142857144"

«block»
ft280 : Fuel_Tank
capacity = "8.0"
current_amount = "7.9"
current_fullness = "0.9875"



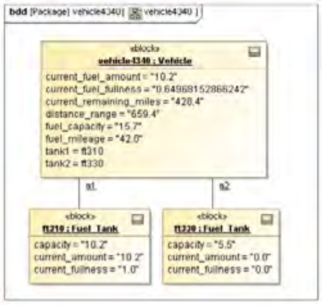
bdd [Package] vehicle4340 [vehicle4340]

«block»
vehicle4340 : Vehicle
current_fuel_amount = "10.2"
current_fuel_fullness = "0.6496815286624"
current_remaining_miles = "428.4"
distance_range = "659.4"
fuel_capacity = "15.7"
fuel_mileage = "42.0"
tank1 = ft310
tank2 = ft330

«block»
ft310 : Fuel_Tank
capacity = "10.2"
current_amount = "10.2"
current_fullness = "1.0"

«block»
ft330 : Fuel_Tank
capacity = "5.5"
current_amount = "0.0"
current_fullness = "0.0"

Exercise 0: Automobile Fuel Capacity & Mileage
Stage 3 Model (p3/3)

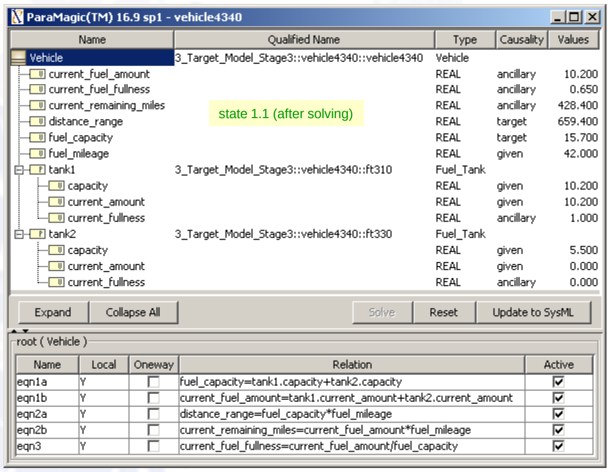


bdd [Package] vehicle4340 [vehicle4340]

«block»
vehicle4340 : Vehicle
current_fuel_amount = "10.2"
current_fuel_fullness = "0.6496815286624"
current_remaining_miles = "428.4"
distance_range = "659.4"
fuel_capacity = "15.7"
fuel_mileage = "42.0"
tank1 = ft310
tank2 = ft330

«block»
ft310 : Fuel_Tank
capacity = "10.2"
current_amount = "10.2"
current_fullness = "1.0"

«block»
ft330 : Fuel_Tank
capacity = "5.5"
current_amount = "0.0"
current_fullness = "0.0"

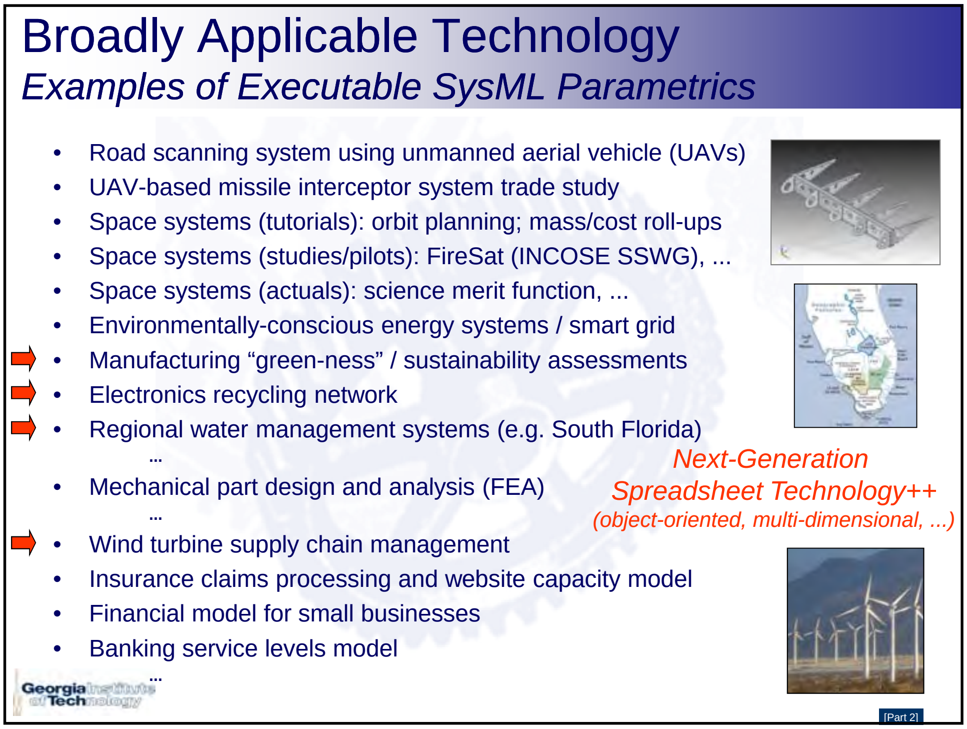


state 1.1 (after solving)

Name	Qualified Name	Type	Causality	Values
Vehicle	3_Target_Model_Stage3::vehicle4340::vehicle4340	Vehicle		
current_fuel_amount		REAL	ancillary	10.200
current_fuel_fullness		REAL	ancillary	0.650
current_remaining_miles		REAL	ancillary	428.400
distance_range		REAL	target	659.400
fuel_capacity		REAL	target	15.700
fuel_mileage		REAL	given	42.000
tank1	3_Target_Model_Stage3::vehicle4340::ft310	Fuel_Tank		
capacity		REAL	given	10.200
current_amount		REAL	given	10.200
current_fullness		REAL	ancillary	1.000
tank2	3_Target_Model_Stage3::vehicle4340::ft330	Fuel_Tank		
capacity		REAL	given	5.500
current_amount		REAL	given	0.000
current_fullness		REAL	ancillary	0.000

root (Vehicle)

Name	Local	Oneway	Relation	Active
eqn1a	Y	☐	fuel_capacity=tank1.capacity+tank2.capacity	☑
eqn1b	Y	☐	current_fuel_amount=tank1.current_amount+tank2.current_amount	☑
eqn2a	Y	☐	distance_range=fuel_capacity*fuel_mileage	☑
eqn2b	Y	☐	current_remaining_miles=current_fuel_amount*fuel_mileage	☑
eqn3	Y	☐	current_fuel_fullness=current_fuel_amount/fuel_capacity	☑



Broadly Applicable Technology

Examples of Executable SysML Parametrics

- Road scanning system using unmanned aerial vehicle (UAVs)
- UAV-based missile interceptor system trade study
- Space systems (tutorials): orbit planning; mass/cost roll-ups
- Space systems (studies/pilots): FireSat (INCOSE SSWG), ...
- Space systems (actuals): science merit function, ...
- Environmentally-conscious energy systems / smart grid
- ➡ Manufacturing “green-ness” / sustainability assessments
- ➡ Electronics recycling network
- ➡ Regional water management systems (e.g. South Florida)

…

- Mechanical part design and analysis (FEA)

…

- ➡ Wind turbine supply chain management
- Insurance claims processing and website capacity model
- Financial model for small businesses
- Banking service levels model

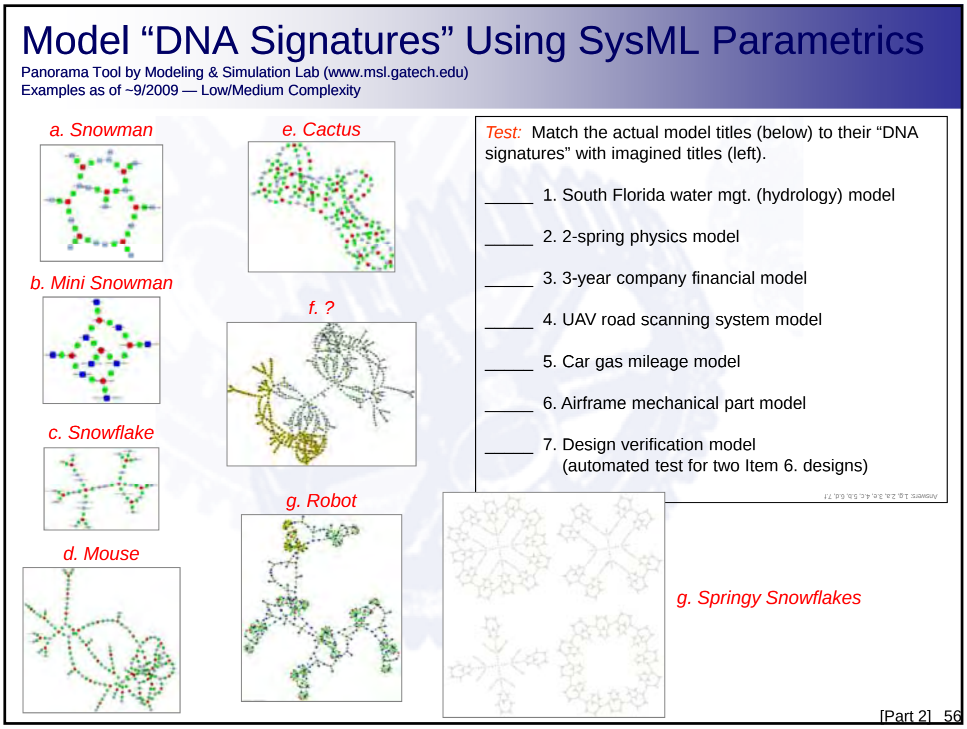
…

Next-Generation Spreadsheet Technology++

(object-oriented, multi-dimensional, ...)

Georgia Institute of Technology

[Part 2]



Model “DNA Signatures” Using SysML Parametrics

Panorama Tool by Modeling & Simulation Lab (www.msl.gatech.edu)
Examples as of ~9/2009 — Low/Medium Complexity

a. Snowman

b. Mini Snowman

c. Snowflake

d. Mouse

e. Cactus

f. ?

g. Robot

g. Springy Snowflakes

Test: Match the actual model titles (below) to their “DNA signatures” with imagined titles (left).

_____ 1. South Florida water mgt. (hydrology) model

_____ 2. 2-spring physics model

_____ 3. 3-year company financial model

_____ 4. UAV road scanning system model

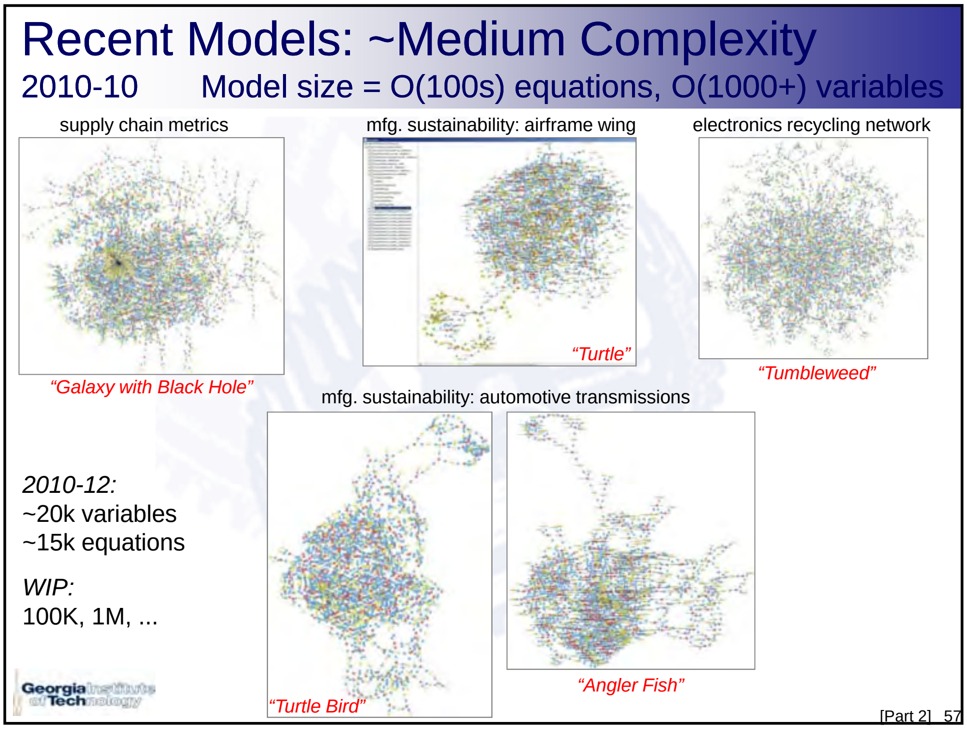
_____ 5. Car gas mileage model

_____ 6. Airframe mechanical part model

_____ 7. Design verification model
(automated test for two Item 6. designs)

Answers: 1.g, 2.a, 3.e, 4.c, 5.b, 6.d, 7.f

[Part 2] 56



Recent Models: ~Medium Complexity

2010-10 Model size = O(100s) equations, O(1000+) variables

supply chain metrics

"Galaxy with Black Hole"

mfg. sustainability: airframe wing

"Turtle"

electronics recycling network

"Tumbleweed"

mfg. sustainability: automotive transmissions

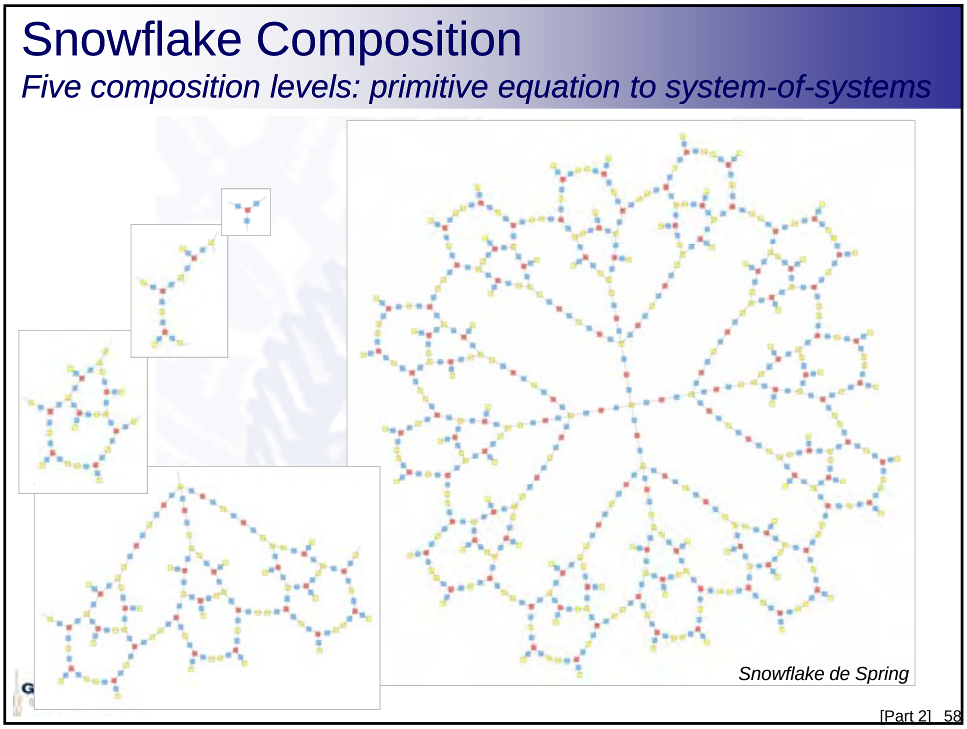
"Turtle Bird"

"Angler Fish"

2010-12:
~20k variables
~15k equations

WIP:
100K, 1M, ...

[Part 2] 57

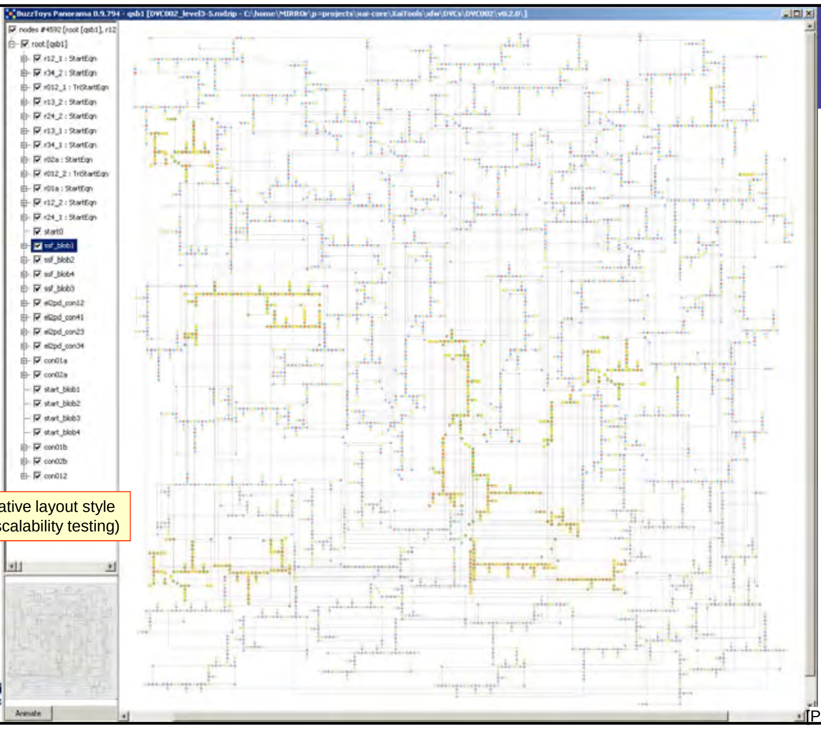


Snowflake Composition

Five composition levels: primitive equation to system-of-systems

Snowflake de Spring

[Part 2] 58

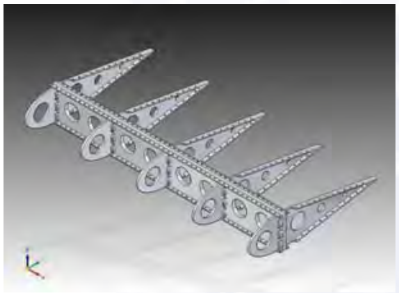


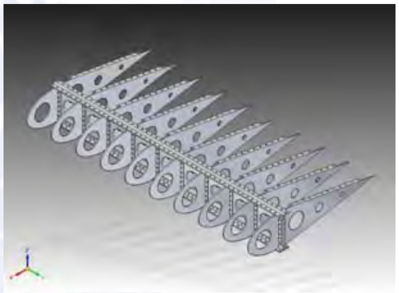
alternative layout style
(and scalability testing)

Using SysML to Evaluate Sustainability Metrics

(similar to Other Metrics: Design Flexibility, ...)

F-86 wing section test case





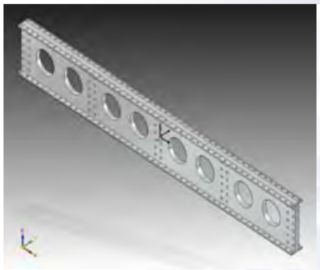
Aluminum Cast and Machined Components
More Room for Internal Parts
Fewer Manufacturing Operations
Heavier

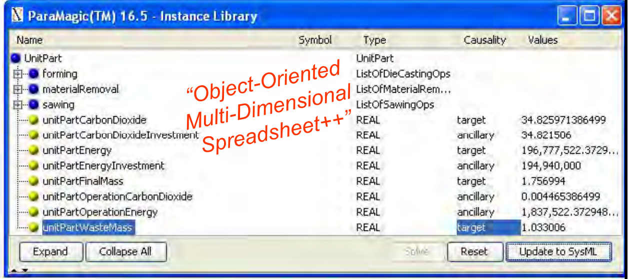
Rolled, Bent, Stamped Sheet Metal
Less Room for Internal Parts
More Manufacturing Operations
Lighter

Source: Bras, Romaniw, et al. 10/2009
www.sdm.gatech.edu

F-86 Wing Section Test Case in SysML Parametrics

Comparing Sustainability Metrics for Design Alternatives



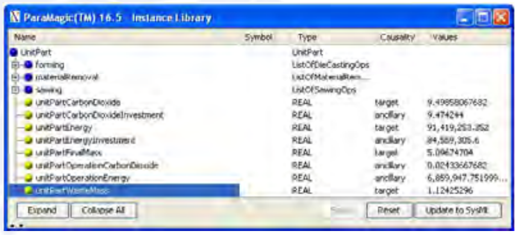


ParaMagic(TM) 16.5 - Instance Library

Name	Symbol	Type	Causality	Values
UnitPart		UnitPart		
forming		ListOfDieCastingOps		
materialRemoval		ListOfMaterialRem...		
sawing		ListOfSawingOps		
unitPartCarbonDioxide		REAL	target	34.825971386499
unitPartCarbonDioxideInvestment		REAL	ancillary	34.821506
unitPartEnergy		REAL	target	196,777,522.3729...
unitPartEnergyInvestment		REAL	ancillary	194,940,000
unitPartFinalMass		REAL	target	1.756994
unitPartOperationCarbonDioxide		REAL	ancillary	0.004465386499
unitPartOperationEnergy		REAL	ancillary	1,837,522.372948...
unitPartWasteMass		REAL	target	1.033006

"Object-Oriented Multi-Dimensional Spreadsheet++"

Value	Description	Change Aluminum to Steel
Total Carbon Dioxide	Total CO2 For Life of Part (up to this stage)	-25.3 kg
Total Energy	Total Energy for Life of Part (up to this stage)	-105.4 MJ = -29.3 kWh
Invested Carbon Dioxide	CO2 in Harvesting/Refining Raw Materials	-25.3 kg
Invested Energy	Energy in Harvesting/Refining Raw Materials	-110.4 MJ = -30.7 kWh
Operation Carbon Dioxide	*Manufacturing/Fabrication CO2*	*+0.02 kg*
Operation Energy	*Manufacturing/Fabrication Energy*	*+502.2 kJ = +1.4 kWh*
Final Mass	Final Part Mass	+3.4 kg
Waste Mass	Total Manufacturing Waste Mass	+0.1 kg



ParaMagic(TM) 16.5 - Instance Library

Name	Symbol	Type	Causality	Values
UnitPart		UnitPart		
forming		ListOfDieCastingOps		
materialRemoval		ListOfMaterialRem...		
sawing		ListOfSawingOps		
unitPartCarbonDioxide		REAL	target	9.49858067682
unitPartCarbonDioxideInvestment		REAL	ancillary	9.474244
unitPartEnergy		REAL	target	91,419,253.252
unitPartEnergyInvestment		REAL	ancillary	84,589,305.6
unitPartFinalMass		REAL	target	5.09674704
unitPartOperationCarbonDioxide		REAL	ancillary	0.02433667682
unitPartOperationEnergy		REAL	ancillary	6,859,947.751999...
unitPartWasteMass		REAL	target	1.12425296

Source: Bras, Romaniw, et al. 10/2009
www.sdm.gatech.edu

[Part 2]

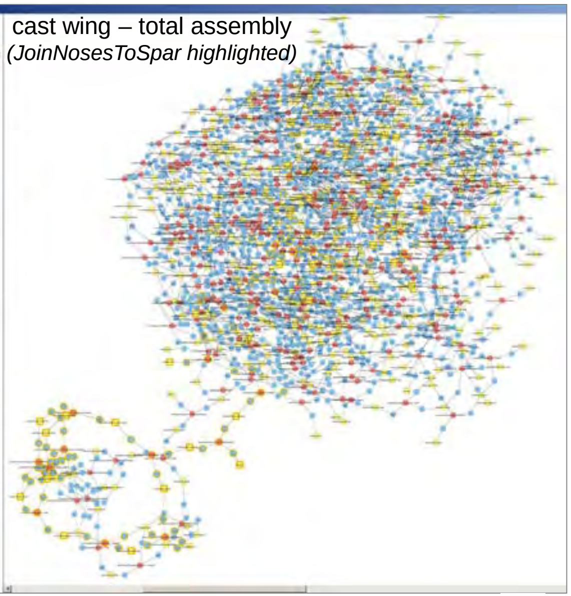
Recent Models: ~Medium Complexity

F-86 Cast Wing Section [adapted from Bras, Romaniw, et al.] – p1/3

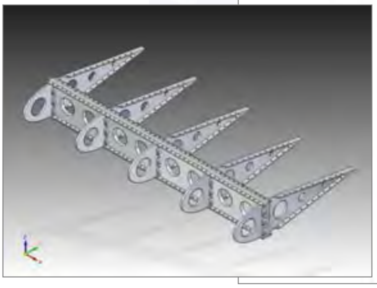
SysML parametrics stats

=== structural stats
23 blocks
218 value properties
38 part properties
0 reference properties
0 shared properties
12 complex aggregate properties
0 primitive properties
195 constraint properties - regular
0 constraint properties - xfwExternal
0 constraint properties - cMathematica

=== instance stats
184 block instances
1879 value property slots
165 part property slots
0 reference property slots
0 shared property slots
53 complex aggregate members
0 primitive aggregate members
346 constraint property eqns - regular
0 constraint property eqns - xfwExternal
0 constraint property eqns - cMathematica



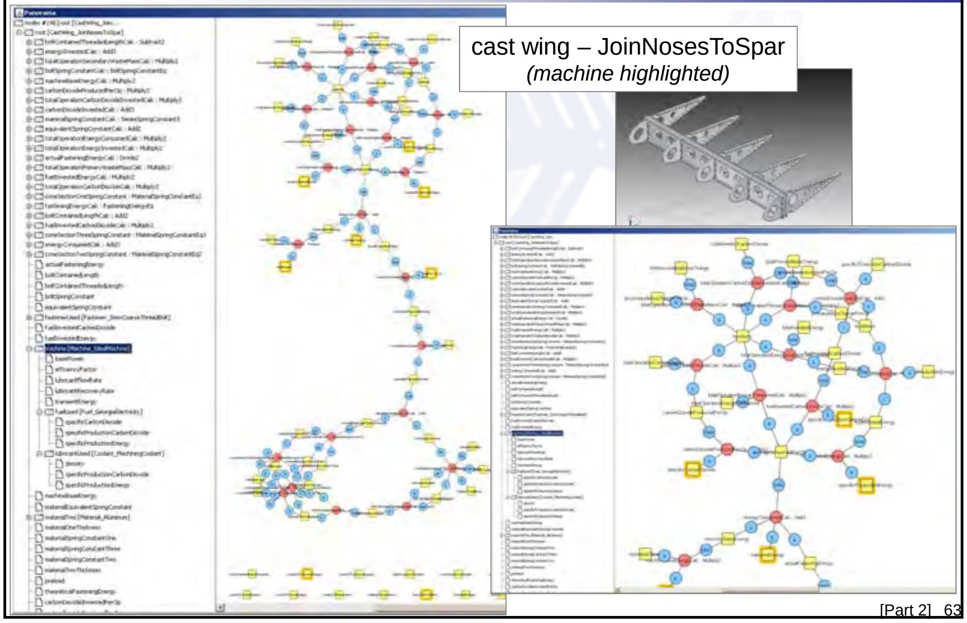
cast wing – total assembly
(JoinNosesToSpar highlighted)



[Part 2] 62

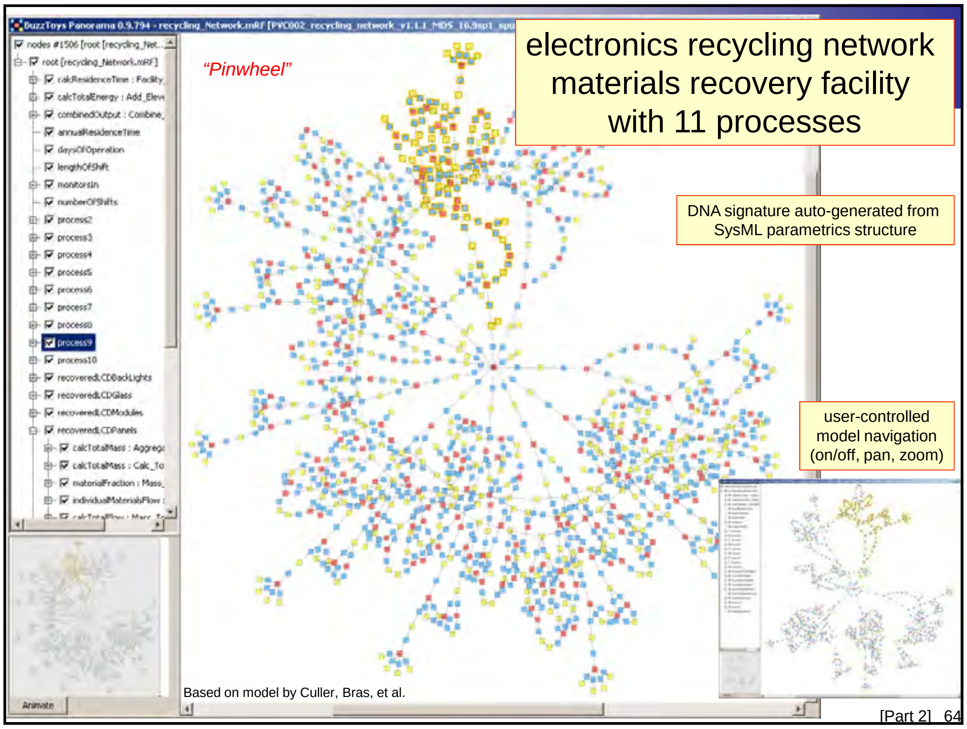
Recent Models: ~Medium Complexity

F-86 Cast Wing Assembly [adapted from Bras, Romaniw, et al.] – p2/3



cast wing – JoinNosesToSpar
(machine highlighted)

[Part 2] 63



"Pinwheel"

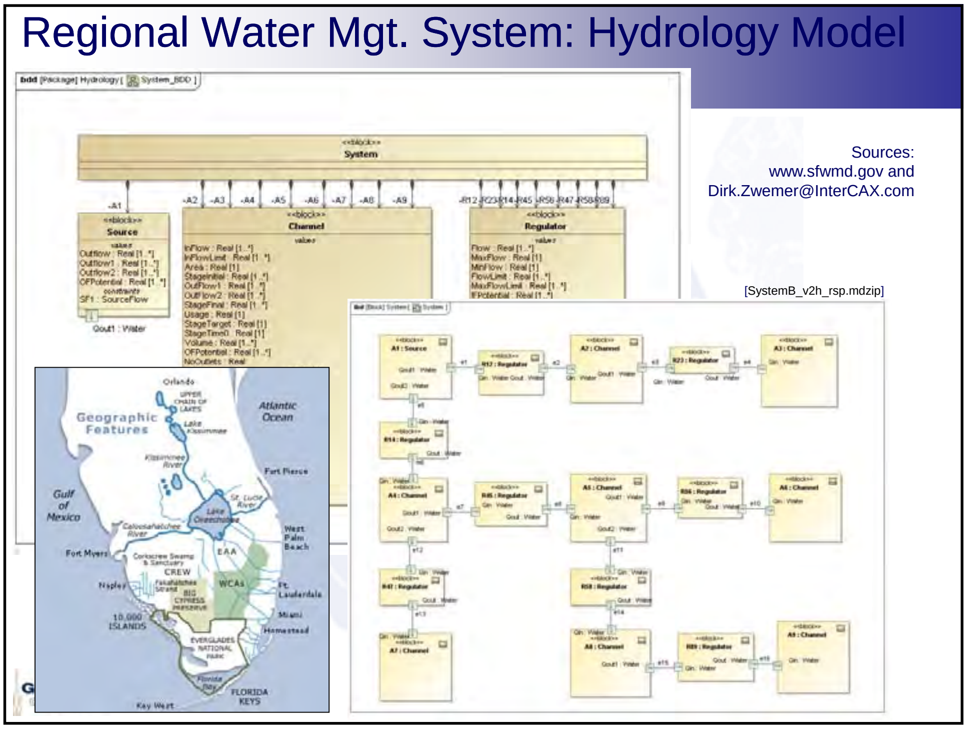
electronics recycling network materials recovery facility with 11 processes

DNA signature auto-generated from SysML parametrics structure

user-controlled model navigation (on/off, pan, zoom)

Based on model by Culler, Bras, et al.

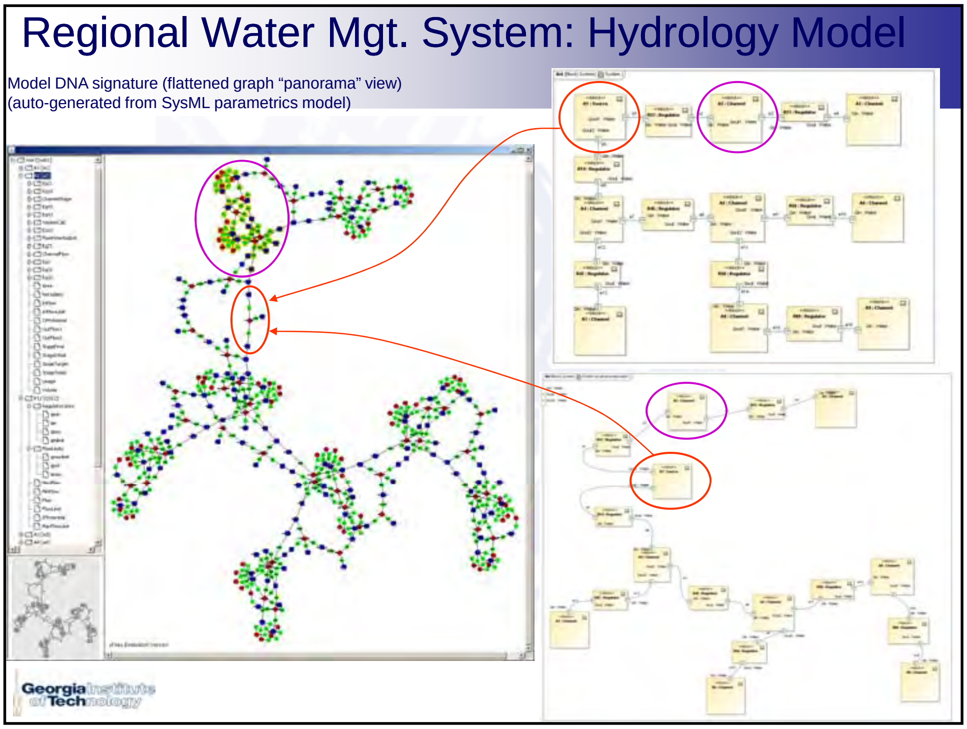
[Part 2] 64



Regional Water Mgt. System: Hydrology Model

Sources:
www.sfwmd.gov and
Dirk.Zwemer@InterCAX.com

[SystemB_v2h_rsp.mdzip]

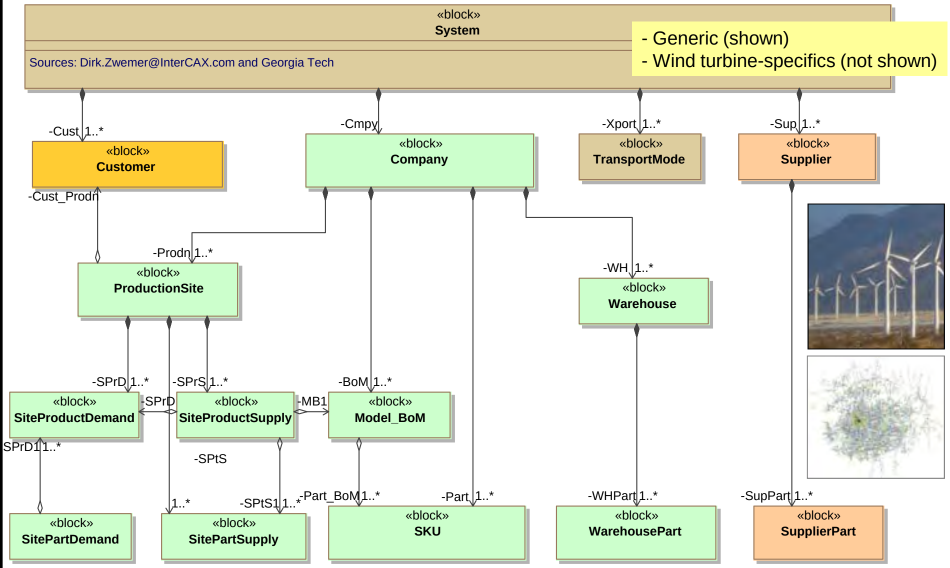


Regional Water Mgt. System: Hydrology Model

Model DNA signature (flattened graph "panorama" view)
(auto-generated from SysML parametrics model)

Supply Chain Model

for Global Supply Chain Management & Optimization



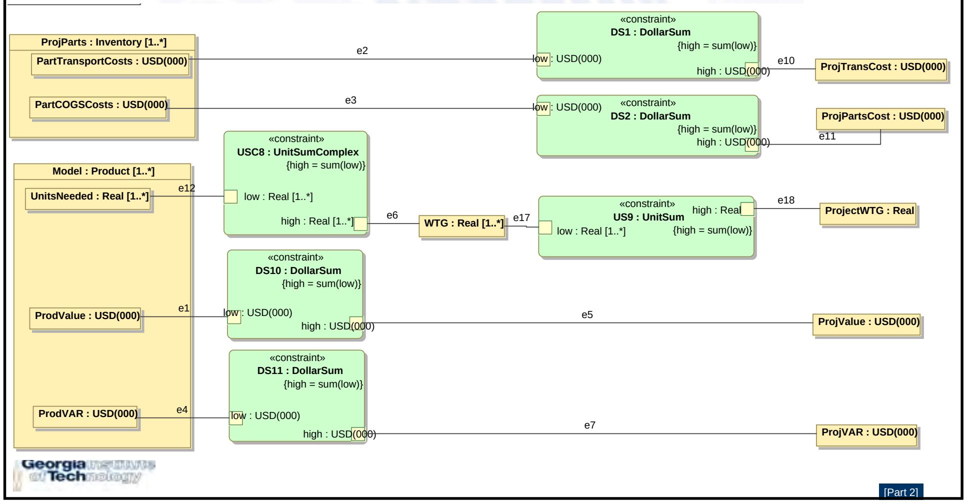
- Generic (shown)
- Wind turbine-specifics (not shown)

[Part 2]

Supply Chain Model – SysML Parametrics

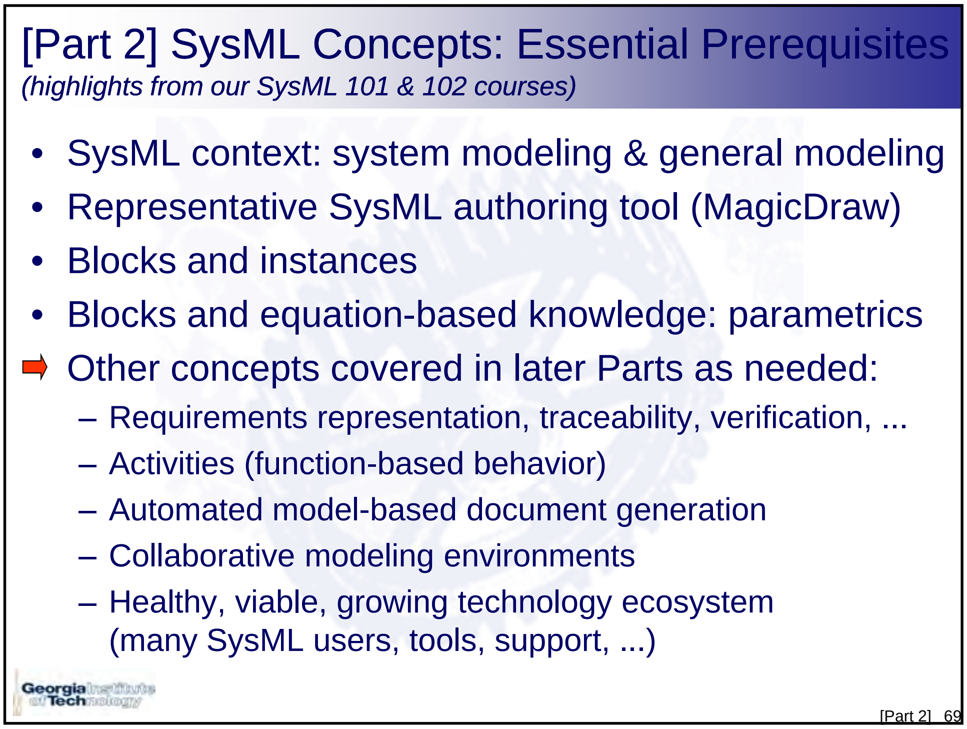
Connect to Optimization Models, Compute Value-at-Risk

Ex. Given 100's of product orders and sourcing plans for the next 12 months, what percent of my business is at-risk if Supplier X does not deliver, or if Part Y becomes obsolete?



Georgia Institute of Technology

[Part 2]



[Part 2] SysML Concepts: Essential Prerequisites

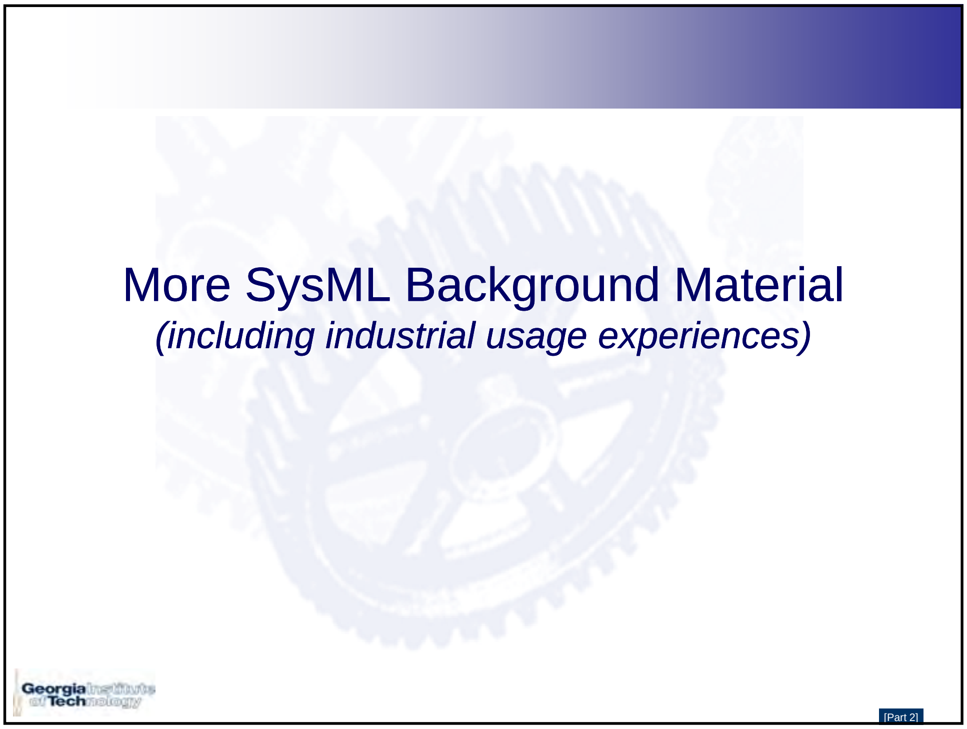
(highlights from our SysML 101 & 102 courses)

- SysML context: system modeling & general modeling
- Representative SysML authoring tool (MagicDraw)
- Blocks and instances
- Blocks and equation-based knowledge: parametrics

➡ Other concepts covered in later Parts as needed:

- Requirements representation, traceability, verification, ...
- Activities (function-based behavior)
- Automated model-based document generation
- Collaborative modeling environments
- Healthy, viable, growing technology ecosystem (many SysML users, tools, support, ...)

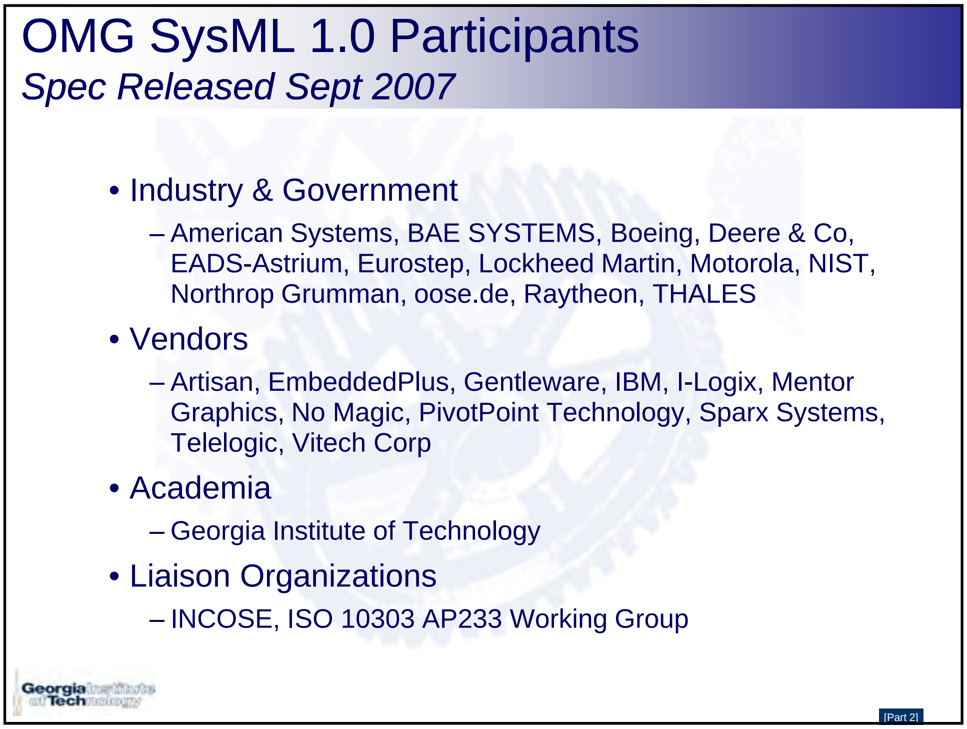
[Part 2] 69



More SysML Background Material

(including industrial usage experiences)

[Part 2]

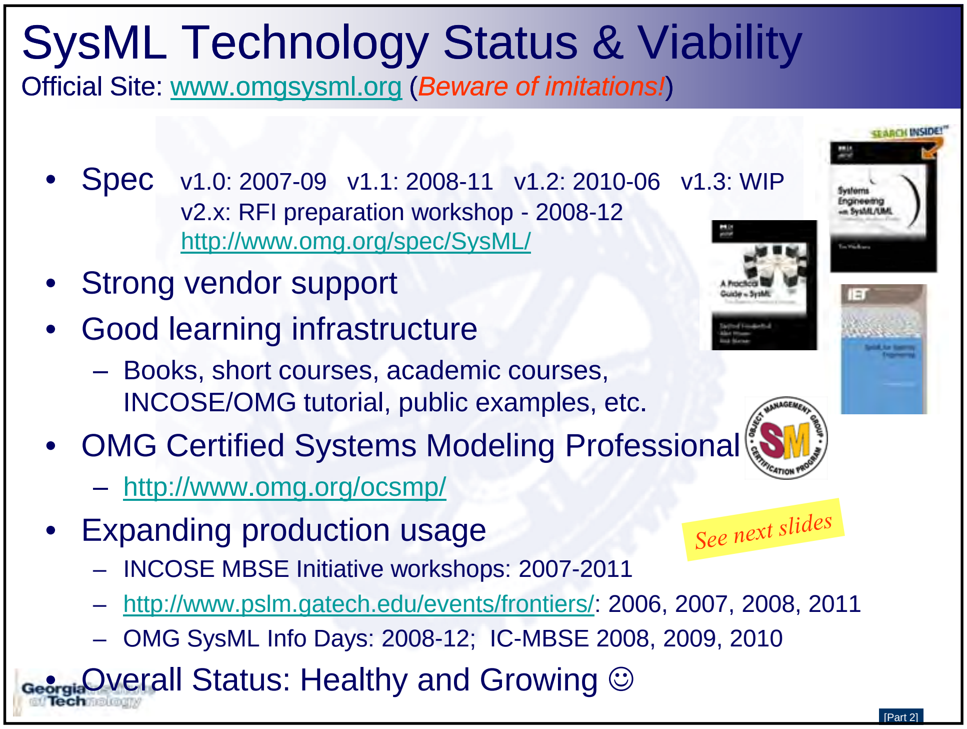


OMG SysML 1.0 Participants

Spec Released Sept 2007

- Industry & Government
  - American Systems, BAE SYSTEMS, Boeing, Deere & Co, EADS-Astrium, Eurostep, Lockheed Martin, Motorola, NIST, Northrop Grumman, oose.de, Raytheon, THALES
- Vendors
  - Artisan, EmbeddedPlus, Gentleware, IBM, I-Logix, Mentor Graphics, No Magic, PivotPoint Technology, Sparx Systems, Telelogic, Vitech Corp
- Academia
  - Georgia Institute of Technology
- Liaison Organizations
  - INCOSE, ISO 10303 AP233 Working Group

[Part 2]



SysML Technology Status & Viability

Official Site: www.omgsysml.org (*Beware of imitations!*)

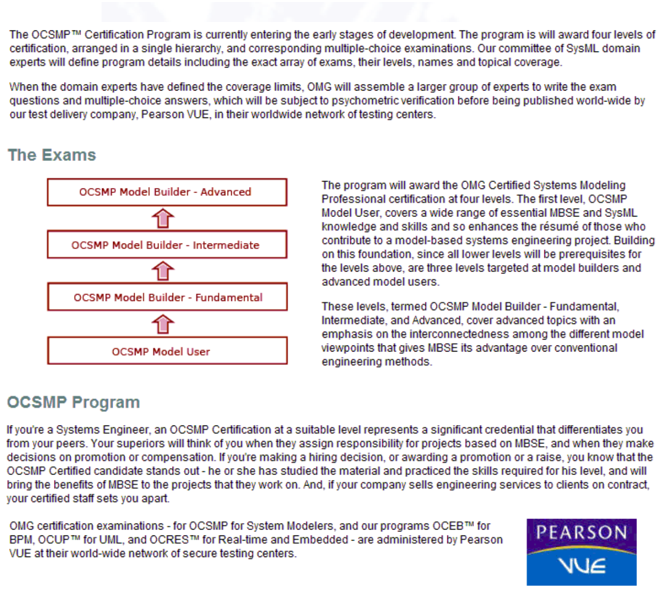
- Spec v1.0: 2007-09 v1.1: 2008-11 v1.2: 2010-06 v1.3: WIP
  v2.x: RFI preparation workshop - 2008-12
  http://www.omg.org/spec/SysML/
- Strong vendor support
- Good learning infrastructure
  - Books, short courses, academic courses, INCOSE/OMG tutorial, public examples, etc.
- OMG Certified Systems Modeling Professional
  - http://www.omg.org/ocsmp/
- Expanding production usage
  - INCOSE MBSE Initiative workshops: 2007-2011
  - http://www.pslm.gatech.edu/events/frontiers/: 2006, 2007, 2008, 2011
  - OMG SysML Info Days: 2008-12; IC-MBSE 2008, 2009, 2010
- Overall Status: Healthy and Growing ☺

See next slides

[Part 2]

OMG Certified Systems Modeling Professional

Certification Program Overview



The OCSMP™ Certification Program is currently entering the early stages of development. The program is will award four levels of certification, arranged in a single hierarchy, and corresponding multiple-choice examinations. Our committee of SysML domain experts will define program details including the exact array of exams, their levels, names and topical coverage.

When the domain experts have defined the coverage limits, OMG will assemble a larger group of experts to write the exam questions and multiple-choice answers, which will be subject to psychometric verification before being published world-wide by our test delivery company, Pearson VUE, in their worldwide network of testing centers.

The Exams

OCSMP Model Builder - Advanced

OCSMP Model Builder - Intermediate

OCSMP Model Builder - Fundamental

OCSMP Model User

The program will award the OMG Certified Systems Modeling Professional certification at four levels. The first level, OCSMP Model User, covers a wide range of essential MBSE and SysML knowledge and skills and so enhances the résumé of those who contribute to a model-based systems engineering project. Building on this foundation, since all lower levels will be prerequisites for the levels above, are three levels targeted at model builders and advanced model users.

These levels, termed OCSMP Model Builder - Fundamental, Intermediate, and Advanced, cover advanced topics with an emphasis on the interconnectedness among the different model viewpoints that gives MBSE its advantage over conventional engineering methods.

OCSMP Program

If you're a Systems Engineer, an OCSMP Certification at a suitable level represents a significant credential that differentiates you from your peers. Your superiors will think of you when they assign responsibility for projects based on MBSE, and when they make decisions on promotion or compensation. If you're making a hiring decision, or awarding a promotion or a raise, you know that the OCSMP Certified candidate stands out - he or she has studied the material and practiced the skills required for his level, and will bring the benefits of MBSE to the projects that they work on. And, if your company sells engineering services to clients on contract, your certified staff sets you apart.

OMG certification examinations - for OCSMP for System Modelers, and our programs OCEB™ for BPM, OCUP™ for UML, and OCRES™ for Real-time and Embedded - are administered by Pearson VUE at their world-wide network of secure testing centers.

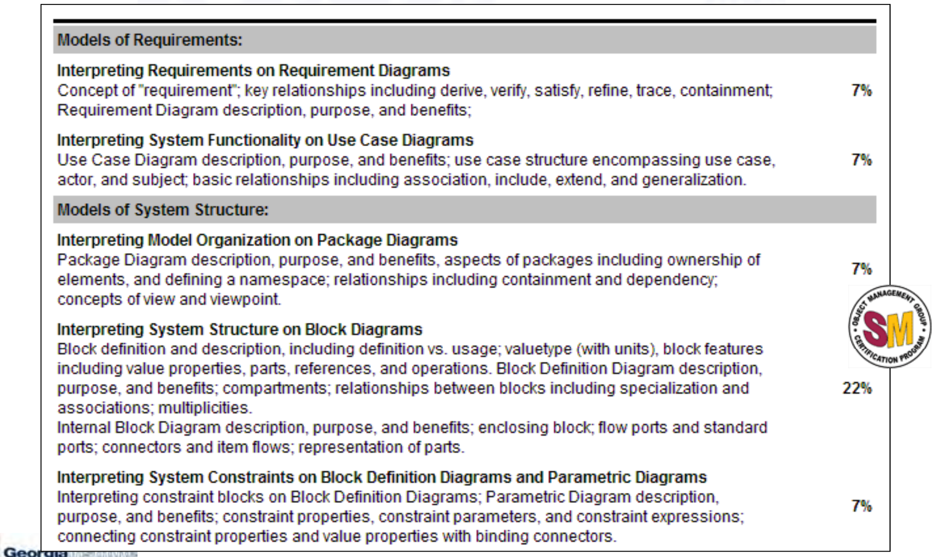
www.omg.org/ocsmp

Status as of Feb 2011
- Beta testing was completed for all Levels 1-4 as of Dec 2010.
- Regular exams are now available for all Levels 1-4.

[Part 2]

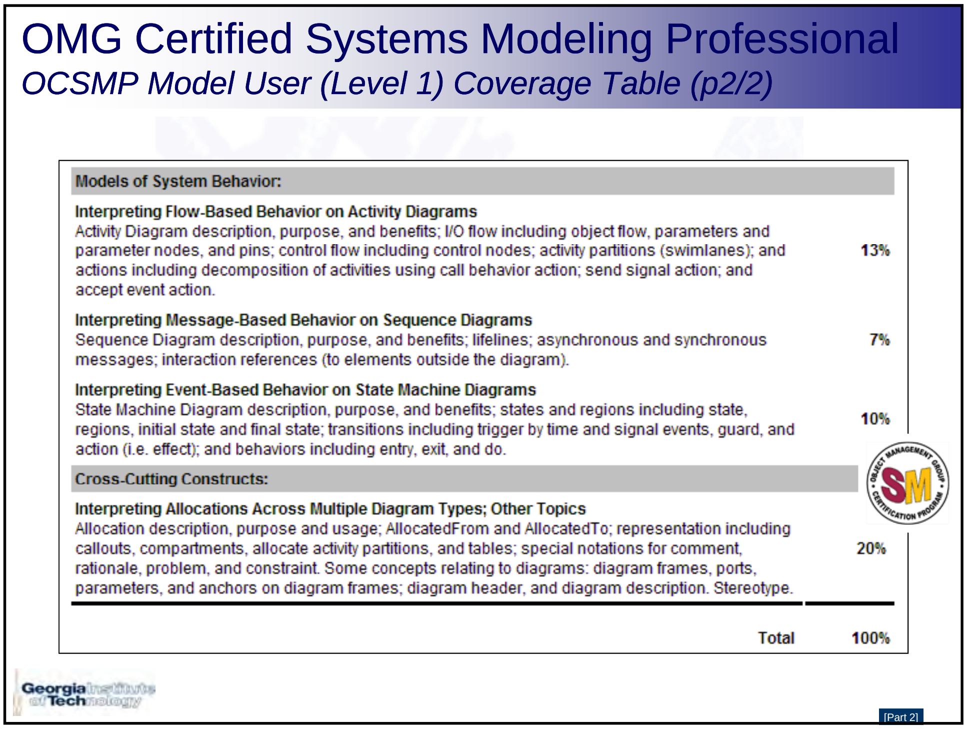
OMG Certified Systems Modeling Professional

OCSMP Model User (Level 1) Coverage Table (p1/2)



Models of Requirements:	
Interpreting Requirements on Requirement Diagrams Concept of "requirement"; key relationships including derive, verify, satisfy, refine, trace, containment; Requirement Diagram description, purpose, and benefits;	7%
Interpreting System Functionality on Use Case Diagrams Use Case Diagram description, purpose, and benefits; use case structure encompassing use case, actor, and subject; basic relationships including association, include, extend, and generalization.	7%
Models of System Structure:	
Interpreting Model Organization on Package Diagrams Package Diagram description, purpose, and benefits, aspects of packages including ownership of elements, and defining a namespace; relationships including containment and dependency; concepts of view and viewpoint.	7%
Interpreting System Structure on Block Diagrams Block definition and description, including definition vs. usage; valuetype (with units), block features including value properties, parts, references, and operations. Block Definition Diagram description, purpose, and benefits; compartments; relationships between blocks including specialization and associations; multiplicities. Internal Block Diagram description, purpose, and benefits; enclosing block; flow ports and standard ports; connectors and item flows; representation of parts.	22%
Interpreting System Constraints on Block Definition Diagrams and Parametric Diagrams Interpreting constraint blocks on Block Definition Diagrams; Parametric Diagram description, purpose, and benefits; constraint properties, constraint parameters, and constraint expressions; connecting constraint properties and value properties with binding connectors.	7%

[Part 2]



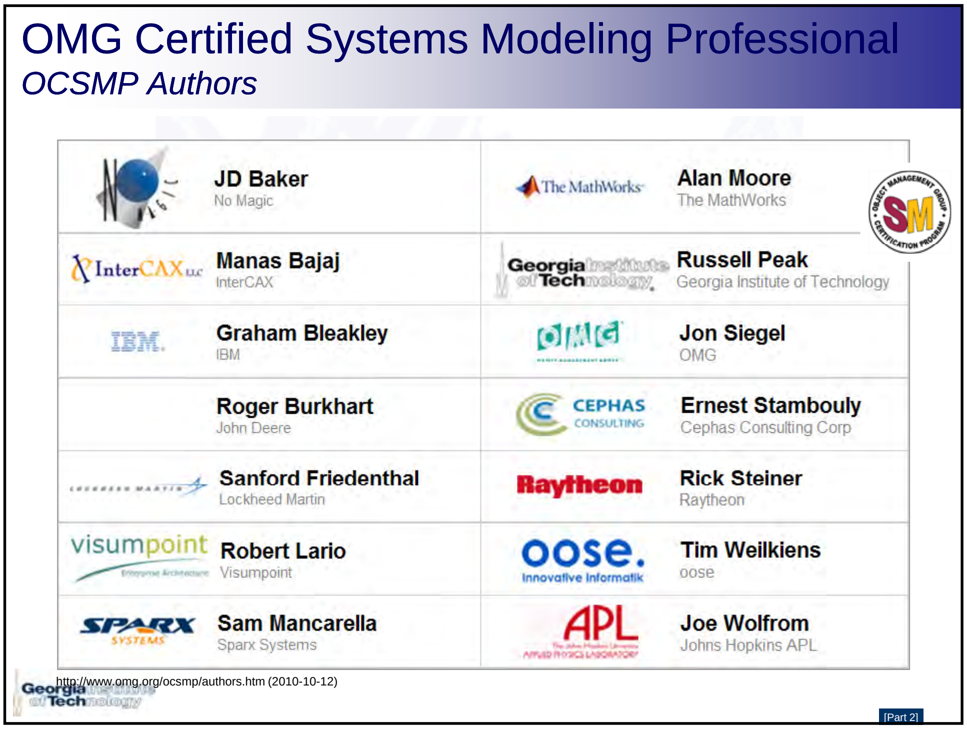
OMG Certified Systems Modeling Professional

OCSMP Model User (Level 1) Coverage Table (p2/2)

Models of System Behavior:	
Interpreting Flow-Based Behavior on Activity Diagrams Activity Diagram description, purpose, and benefits; I/O flow including object flow, parameters and parameter nodes, and pins; control flow including control nodes; activity partitions (swimlanes); and actions including decomposition of activities using call behavior action; send signal action; and accept event action.	13%
Interpreting Message-Based Behavior on Sequence Diagrams Sequence Diagram description, purpose, and benefits; lifelines; asynchronous and synchronous messages; interaction references (to elements outside the diagram).	7%
Interpreting Event-Based Behavior on State Machine Diagrams State Machine Diagram description, purpose, and benefits; states and regions including state, regions, initial state and final state; transitions including trigger by time and signal events, guard, and action (i.e. effect); and behaviors including entry, exit, and do.	10%
Cross-Cutting Constructs:	
Interpreting Allocations Across Multiple Diagram Types; Other Topics Allocation description, purpose and usage; AllocatedFrom and AllocatedTo; representation including callouts, compartments, allocate activity partitions, and tables; special notations for comment, rationale, problem, and constraint. Some concepts relating to diagrams: diagram frames, ports, parameters, and anchors on diagram frames; diagram header, and diagram description. Stereotype.	20%
Total	**100%**

Georgia Institute of Technology

[Part 2]



OMG Certified Systems Modeling Professional

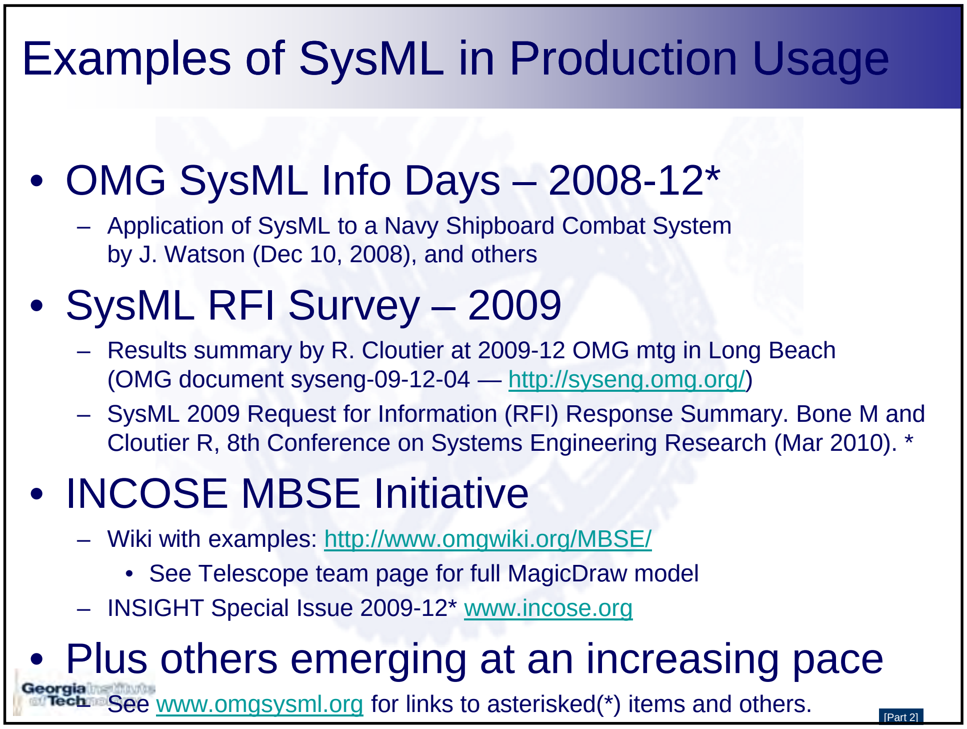
OCSMP Authors

JD Baker No Magic	**Alan Moore** The MathWorks
Manas Bajaj InterCAX	**Russell Peak** Georgia Institute of Technology
Graham Bleakley IBM	**Jon Siegel** OMG
Roger Burkhart John Deere	**Ernest Stambouly** Cephas Consulting Corp
Sanford Friedenthal Lockheed Martin	**Rick Steiner** Raytheon
Robert Lario Visumpoint	**Tim Weilkiens** oose
Sam Mancarella Sparx Systems	**Joe Wolfrom** Johns Hopkins APL

http://www.omg.org/ocsmp/authors.htm (2010-10-12)

Georgia Institute of Technology

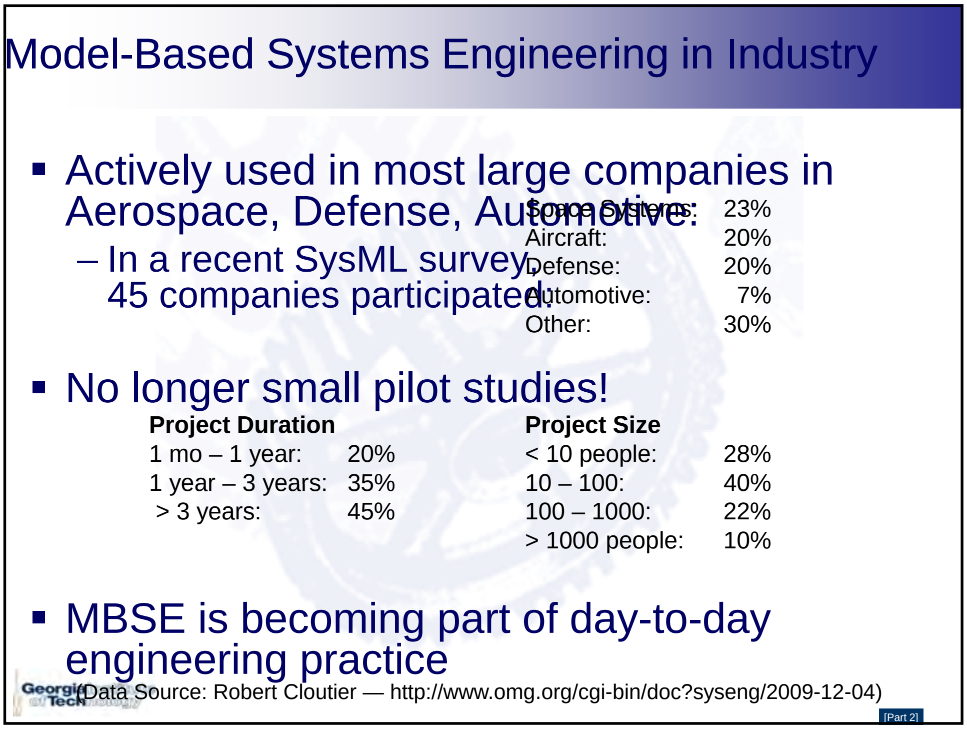
[Part 2]



Examples of SysML in Production Usage

- OMG SysML Info Days – 2008-12*
  - Application of SysML to a Navy Shipboard Combat System by J. Watson (Dec 10, 2008), and others
- SysML RFI Survey – 2009
  - Results summary by R. Cloutier at 2009-12 OMG mtg in Long Beach (OMG document syseng-09-12-04 — http://syseng.omg.org/)
  - SysML 2009 Request for Information (RFI) Response Summary. Bone M and Cloutier R, 8th Conference on Systems Engineering Research (Mar 2010). *
- INCOSE MBSE Initiative
  - Wiki with examples: http://www.omgwiki.org/MBSE/
    - See Telescope team page for full MagicDraw model
  - INSIGHT Special Issue 2009-12* www.incose.org
- Plus others emerging at an increasing pace
  - See www.omgsysml.org for links to asterisked(*) items and others.

[Part 2]



Model-Based Systems Engineering in Industry

- Actively used in most large companies in Aerospace, Defense, Automotive:
  - In a recent SysML survey, 45 companies participated:

Space Systems:	23%
Aircraft:	20%
Defense:	20%
Automotive:	7%
Other:	30%

- No longer small pilot studies!

Project Duration	
1 mo – 1 year:	20%
1 year – 3 years:	35%
> 3 years:	45%

Project Size	
< 10 people:	28%
10 – 100:	40%
100 – 1000:	22%
> 1000 people:	10%

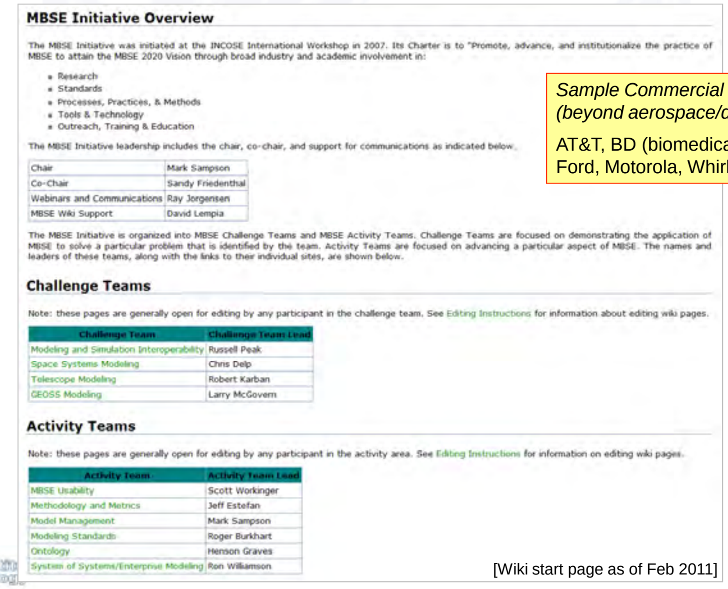
- MBSE is becoming part of day-to-day engineering practice

(Data Source: Robert Cloutier — http://www.omg.org/cgi-bin/doc?syseng/2009-12-04)

[Part 2]

INCOSE MBSE Initiative

http://www.omgwiki.org/MBSE/



MBSE Initiative Overview

The MBSE Initiative was initiated at the INCOSE International Workshop in 2007. Its Charter is to "Promote, advance, and institutionalize the practice of MBSE to attain the MBSE 2020 Vision through broad industry and academic involvement in:

- Research
- Standards
- Processes, Practices, & Methods
- Tools & Technology
- Outreach, Training & Education

The MBSE Initiative leadership includes the chair, co-chair, and support for communications as indicated below.

Chair	Mark Sampson
Co-Chair	Sandy Friedenthal
Webinars and Communications	Ray Jorgensen
MBSE Wiki Support	David Lempia

The MBSE Initiative is organized into MBSE Challenge Teams and MBSE Activity Teams. Challenge Teams are focused on demonstrating the application of MBSE to solve a particular problem that is identified by the team. Activity Teams are focused on advancing a particular aspect of MBSE. The names and leaders of these teams, along with the links to their individual sites, are shown below.

Challenge Teams

Note: these pages are generally open for editing by any participant in the challenge team. See Editing Instructions for information about editing wiki pages.

Challenge Team	Challenge Team Lead
Modeling and Simulation Interoperability	Russell Peak
Space Systems Modeling	Chris Delp
Telescope Modeling	Robert Karban
GEOSS Modeling	Larry McGovern

Activity Teams

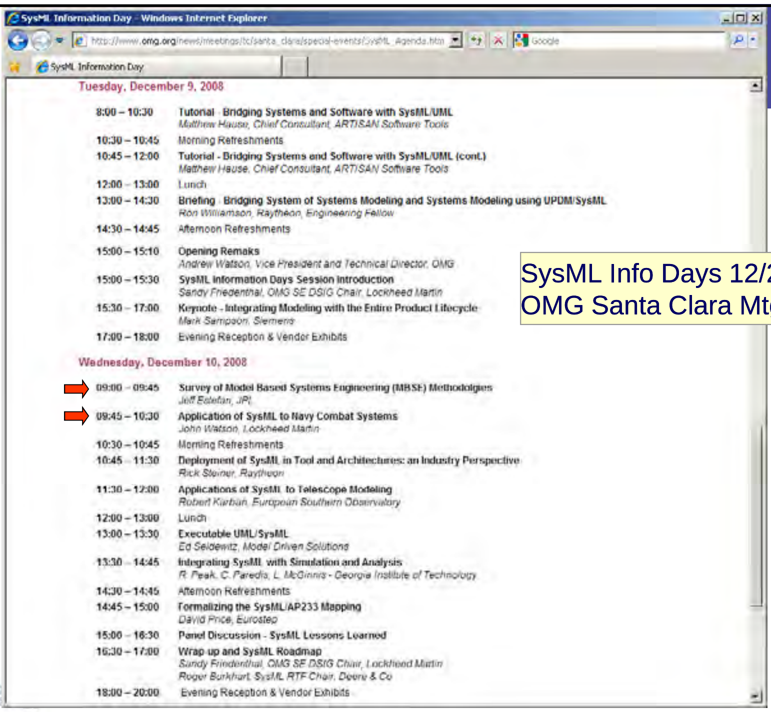
Note: these pages are generally open for editing by any participant in the activity area. See Editing Instructions for information on editing wiki pages.

Activity Team	Activity Team Lead
MBSE Usability	Scott Workinger
Methodology and Metrics	Jeff Estefan
Model Management	Mark Sampson
Modeling Standards	Roger Burkhart
Ontology	Henson Graves
System of Systems/Enterprise Modeling	Ron Williamson

Sample Commercial Companies (beyond aerospace/defense):

AT&T, BD (biomedical), Deere, Ford, Motorola, Whirlpool, ...

[Wiki start page as of Feb 2011]



SysML Information Day - Windows Internet Explorer
http://www.omg.org/news/meetings/tc/santa_clara/special-events/SysML_Agenda.htm
Google
SysML Information Day

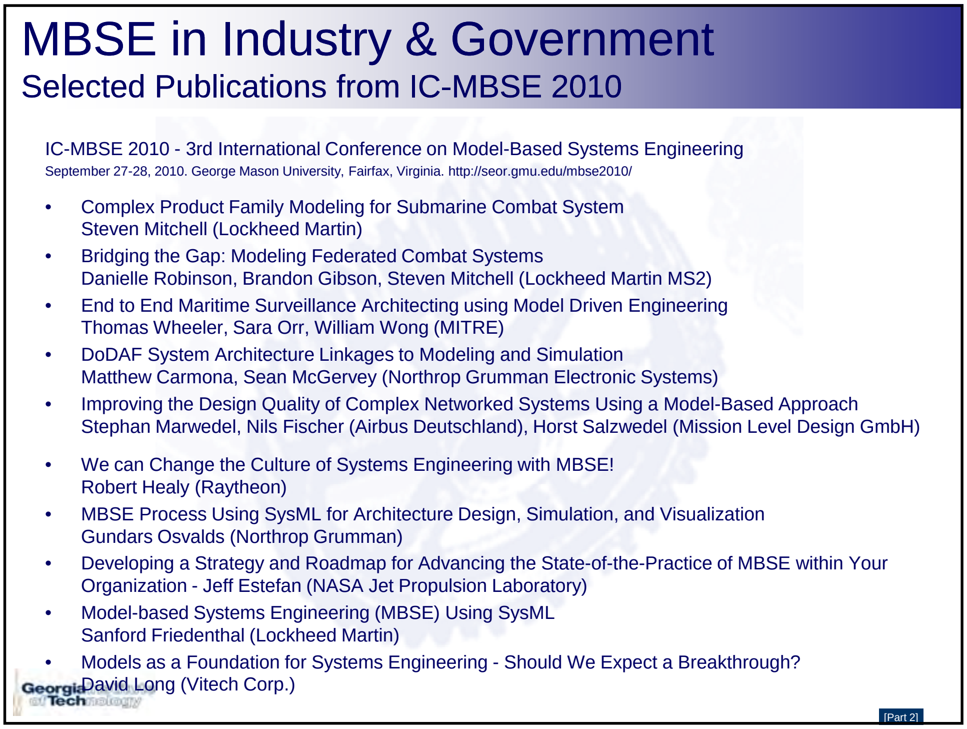
Tuesday, December 9, 2008

8:00 – 10:30	**Tutorial - Bridging Systems and Software with SysML/UML** *Matthew Hause, Chief Consultant, ARTISAN Software Tools*
10:30 – 10:45	Morning Refreshments
10:45 – 12:00	**Tutorial - Bridging Systems and Software with SysML/UML (cont.)** *Matthew Hause, Chief Consultant, ARTISAN Software Tools*
12:00 – 13:00	Lunch
13:00 – 14:30	**Briefing - Bridging System of Systems Modeling and Systems Modeling using UPDM/SysML** *Ron Williamson, Raytheon, Engineering Fellow*
14:30 – 14:45	Afternoon Refreshments
15:00 – 15:10	**Opening Remaks** *Andrew Watson, Vice President and Technical Director, OMG*
15:00 – 15:30	**SysML Information Days Session Introduction** *Sandy Friedenthal, OMG SE DSIG Chair, Lockheed Martin*
15:30 – 17:00	**Keynote - Integrating Modeling with the Entire Product Lifecycle** *Mark Sampson, Siemens*
17:00 – 18:00	Evening Reception & Vendor Exhibits

Wednesday, December 10, 2008

09:00 – 09:45	**Survey of Model Based Systems Engineering (MBSE) Methodolgies** *Jeff Estefan, JPL*
09:45 – 10:30	**Application of SysML to Navy Combat Systems** *John Watson, Lockheed Martin*
10:30 – 10:45	Morning Refreshments
10:45 – 11:30	**Deployment of SysML in Tool and Architectures: an Industry Perspective** *Rick Steiner, Raytheon*
11:30 – 12:00	**Applications of SysML to Telescope Modeling** *Robert Karban, European Southern Observatory*
12:00 – 13:00	Lunch
13:00 – 13:30	**Executable UML/SysML** *Ed Seidewitz, Model Driven Solutions*
13:30 – 14:45	**Integrating SysML with Simulation and Analysis** *R. Peak, C. Paredis, L. McGinnis - Georgia Institute of Technology*
14:30 – 14:45	Afternoon Refreshments
14:45 – 15:00	**Formalizing the SysML/AP233 Mapping** *David Price, Eurostep*
15:00 – 16:30	**Panel Discussion - SysML Lessons Learned**
16:30 – 17:00	**Wrap up and SysML Roadmap** *Sandy Friedenthal, OMG SE DSIG Chair, Lockheed Martin* *Roger Burkhart, SysML RTF Chair, Deere & Co*
18:00 – 20:00	Evening Reception & Vendor Exhibits

SysML Info Days 12/2008
OMG Santa Clara Mtg



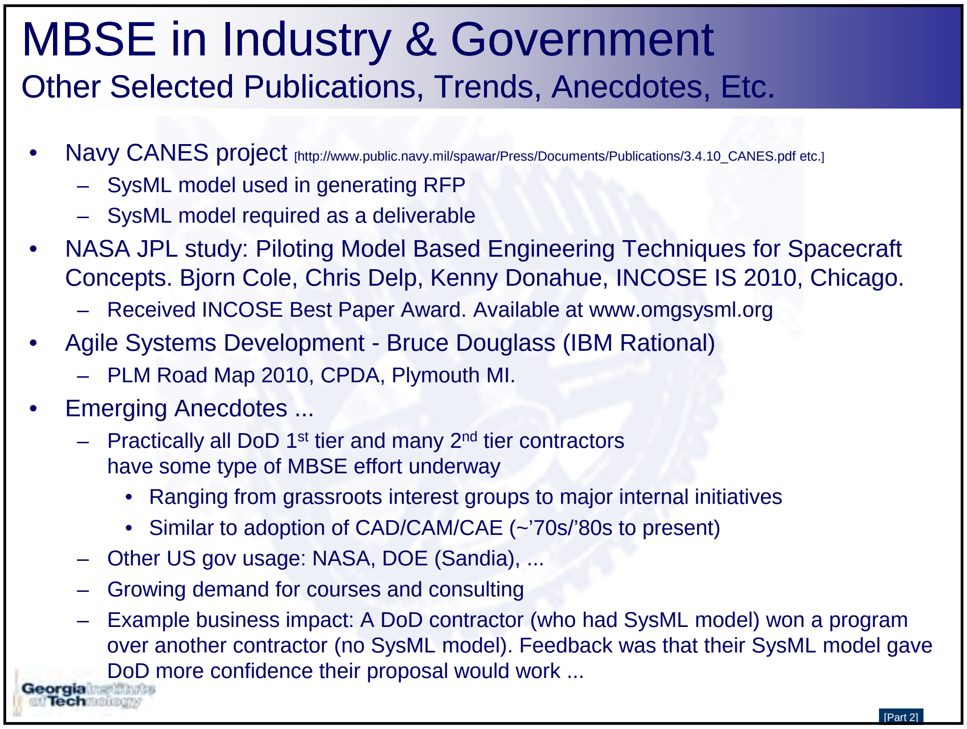
MBSE in Industry & Government

Selected Publications from IC-MBSE 2010

IC-MBSE 2010 - 3rd International Conference on Model-Based Systems Engineering
September 27-28, 2010. George Mason University, Fairfax, Virginia. http://seor.gmu.edu/mbse2010/

- Complex Product Family Modeling for Submarine Combat System
  Steven Mitchell (Lockheed Martin)
- Bridging the Gap: Modeling Federated Combat Systems
  Danielle Robinson, Brandon Gibson, Steven Mitchell (Lockheed Martin MS2)
- End to End Maritime Surveillance Architecting using Model Driven Engineering
  Thomas Wheeler, Sara Orr, William Wong (MITRE)
- DoDAF System Architecture Linkages to Modeling and Simulation
  Matthew Carmona, Sean McGervey (Northrop Grumman Electronic Systems)
- Improving the Design Quality of Complex Networked Systems Using a Model-Based Approach
  Stephan Marwedel, Nils Fischer (Airbus Deutschland), Horst Salzwedel (Mission Level Design GmbH)
- We can Change the Culture of Systems Engineering with MBSE!
  Robert Healy (Raytheon)
- MBSE Process Using SysML for Architecture Design, Simulation, and Visualization
  Gundars Osvalds (Northrop Grumman)
- Developing a Strategy and Roadmap for Advancing the State-of-the-Practice of MBSE within Your Organization - Jeff Estefan (NASA Jet Propulsion Laboratory)
- Model-based Systems Engineering (MBSE) Using SysML
  Sanford Friedenthal (Lockheed Martin)
- Models as a Foundation for Systems Engineering - Should We Expect a Breakthrough?
  David Long (Vitech Corp.)

[Part 2]

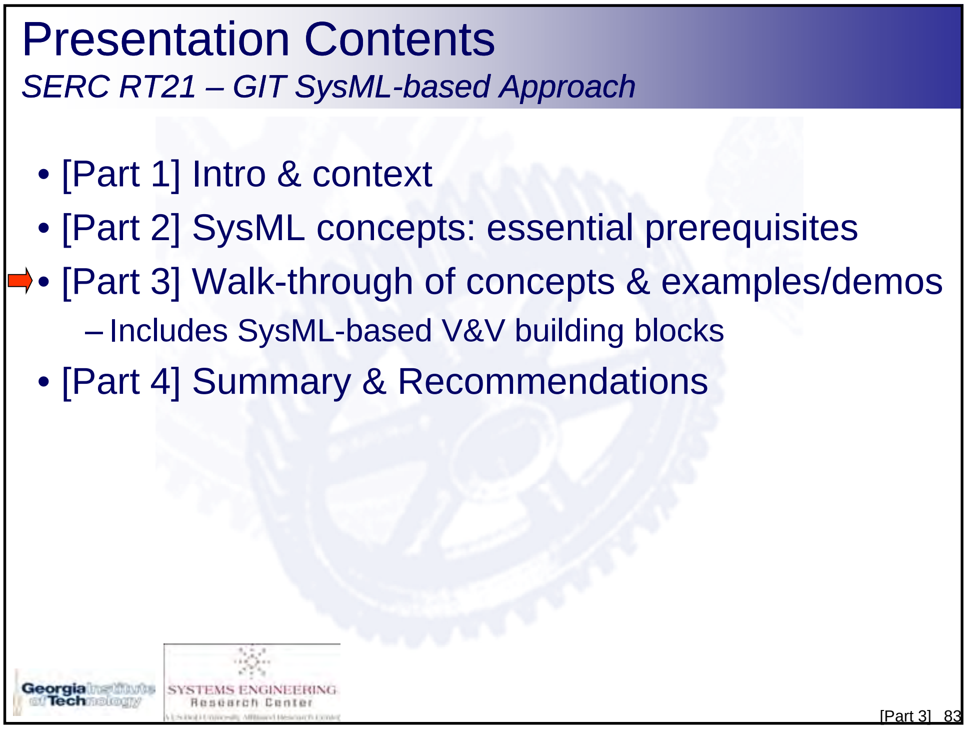


MBSE in Industry & Government

Other Selected Publications, Trends, Anecdotes, Etc.

- Navy CANES project [http://www.public.navy.mil/spawar/Press/Documents/Publications/3.4.10_CANES.pdf etc.]
  - SysML model used in generating RFP
  - SysML model required as a deliverable
- NASA JPL study: Piloting Model Based Engineering Techniques for Spacecraft Concepts. Bjorn Cole, Chris Delp, Kenny Donahue, INCOSE IS 2010, Chicago.
  - Received INCOSE Best Paper Award. Available at www.omgsysml.org
- Agile Systems Development - Bruce Douglass (IBM Rational)
  - PLM Road Map 2010, CPDA, Plymouth MI.
- Emerging Anecdotes ...
  - Practically all DoD 1st tier and many 2nd tier contractors have some type of MBSE effort underway
    - Ranging from grassroots interest groups to major internal initiatives
    - Similar to adoption of CAD/CAM/CAE (~'70s/'80s to present)
  - Other US gov usage: NASA, DOE (Sandia), ...
  - Growing demand for courses and consulting
  - Example business impact: A DoD contractor (who had SysML model) won a program over another contractor (no SysML model). Feedback was that their SysML model gave DoD more confidence their proposal would work ...

[Part 2]

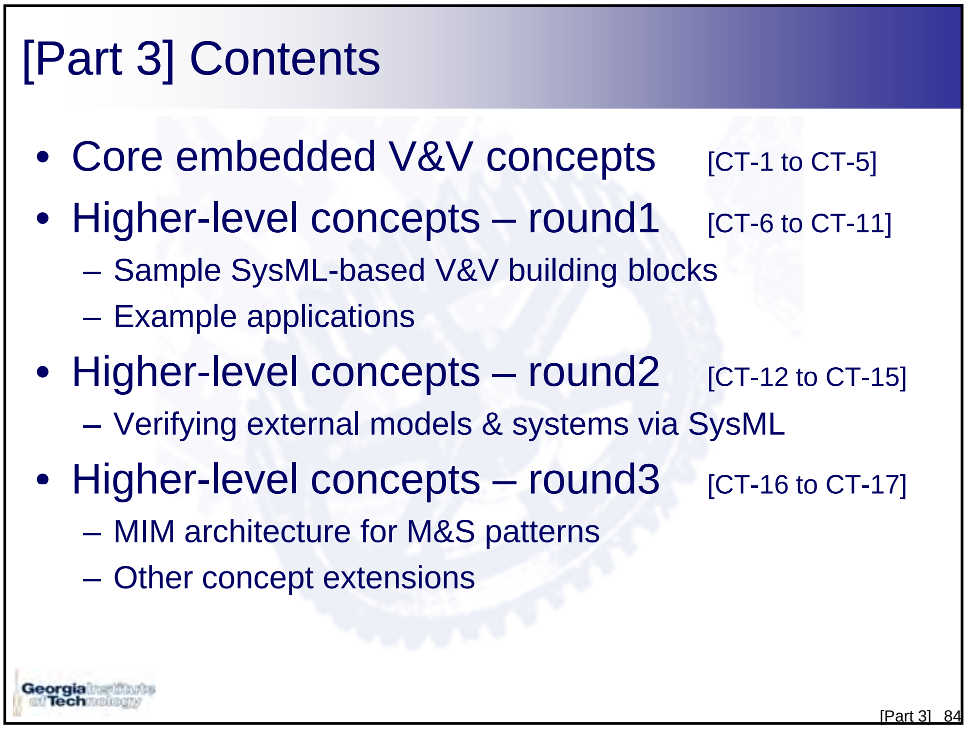


Presentation Contents

SERC RT21 – GIT SysML-based Approach

- [Part 1] Intro & context
- [Part 2] SysML concepts: essential prerequisites
- ➡ [Part 3] Walk-through of concepts & examples/demos
  - Includes SysML-based V&V building blocks
- [Part 4] Summary & Recommendations

[Part 3] 83

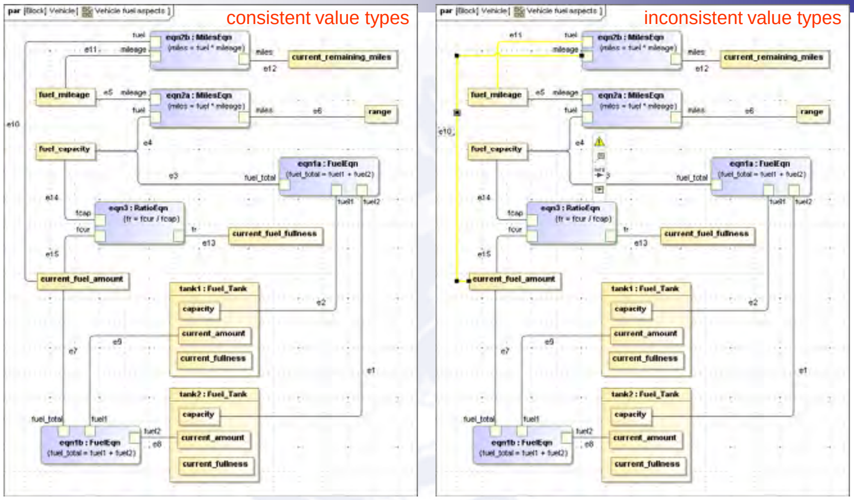


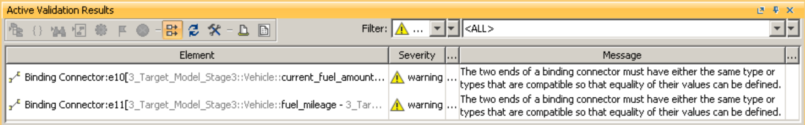
[Part 3] Contents

- Core embedded V&V concepts [CT-1 to CT-5]
- Higher-level concepts – round1 [CT-6 to CT-11]
  - Sample SysML-based V&V building blocks
  - Example applications
- Higher-level concepts – round2 [CT-12 to CT-15]
  - Verifying external models & systems via SysML
- Higher-level concepts – round3 [CT-16 to CT-17]
  - MIM architecture for M&S patterns
  - Other concept extensions

[Part 3] 84

CT-1: Automated units consistency checking



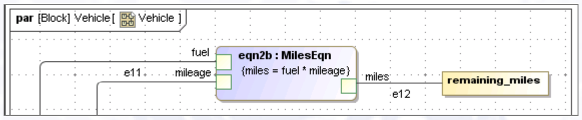


Element	Severity	...	Message
Binding Connector:e10[3_Target_Model_Stage3::Vehicle::current_fuel_amount…	⚠ warning	…	The two ends of a binding connector must have either the same type or types that are compatible so that equality of their values can be defined.
Binding Connector:e11[3_Target_Model_Stage3::Vehicle::fuel_mileage - 3_Tar…	⚠ warning	…	The two ends of a binding connector must have either the same type or types that are compatible so that equality of their values can be defined.

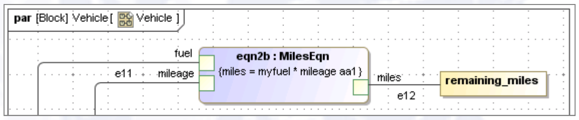
CT-1-1

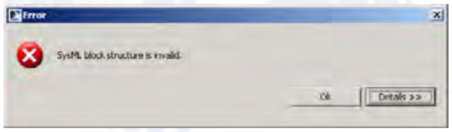
CT-2: Automated equation checking

valid constraint expression



invalid constraint expression


par [Block] Vehicle [Vehicle]
fuel
e11
mileage
eqn2b : MilesEqn
{miles = myfuel * mileage aa1 }
miles
e12
remaining_miles



SysML block structure is invalid.

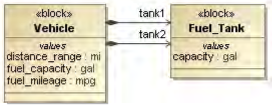
Parsing Failed. expression: miles = myfuel * mileage aa1 has syntax errors caused by aa1 and myfuel.

CT-2-1

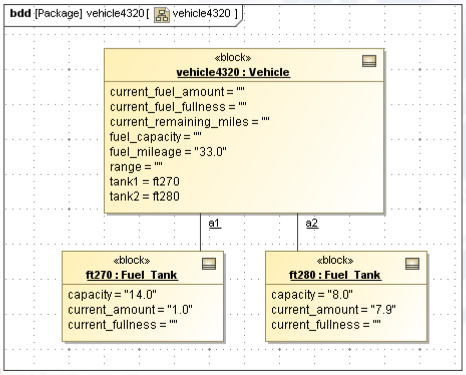
CT-3.1: Model integrity checking

instance consistency with element definitions

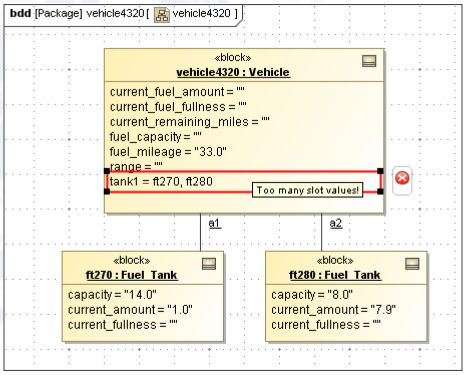
element definitions



valid instance



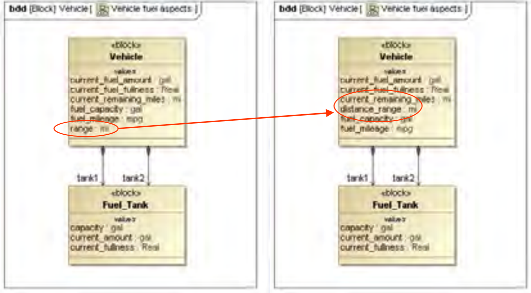
invalid instance: tank1 should have only one value



CT-3-1

CT-3.2: Propagating model updates

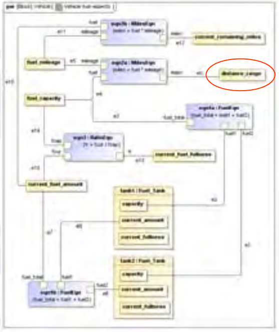
Renaming an element auto-updates all occurrences throughout the model

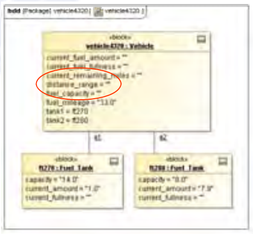


original model – bdd view

updated model – bdd view

updated model – other views (auto-updated)

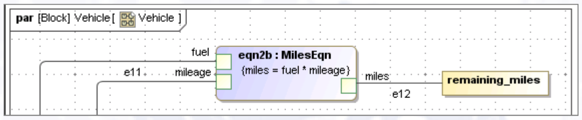




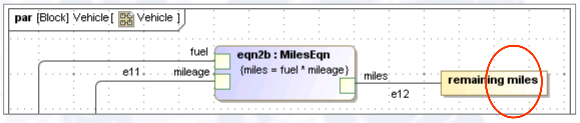
CT-3-2

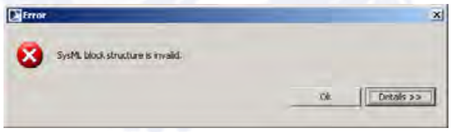
CT-4: Augmented language-level integrity: ensuring best practices, etc.

recommended practice for property names: having no spaces



non-recommended practice for property names: having spaces



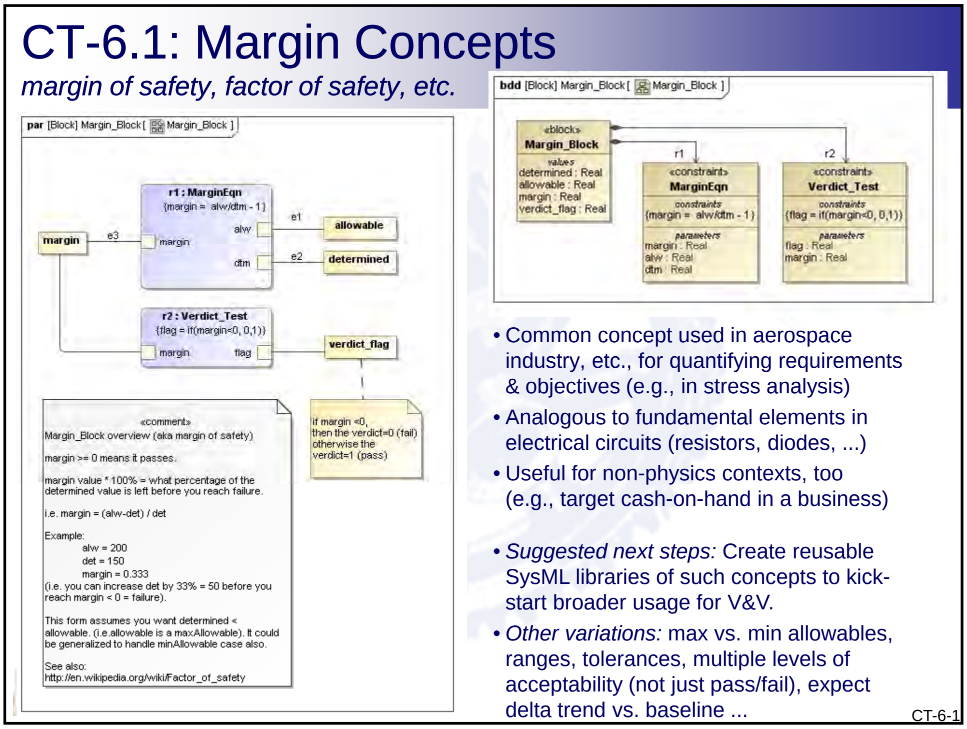


SysML block structure is invalid.

Property name “remaining miles” contains one or more non-recommended character(s).

[Part 3] Contents

- Core embedded V&V concepts [CT-1 to CT-5]
- ➡ Higher-level concepts – round1 [CT-6 to CT-11]
  - Sample SysML-based V&V building blocks
  - Example applications
- Higher-level concepts – round2 [CT-12 to CT-15]
  - Verifying external models & systems via SysML
- Higher-level concepts – round3 [CT-16 to CT-17]
  - MIM architecture for M&S patterns
  - Other concept extensions



CT-6.1: Margin Concepts

margin of safety, factor of safety, etc.

«comment»
Margin_Block overview (aka margin of safety)

margin >= 0 means it passes.

margin value * 100% = what percentage of the determined value is left before you reach failure.

i.e. margin = (alw-det) / det

Example:
alw = 200
det = 150
margin = 0.333
(i.e. you can increase det by 33% = 50 before you reach margin < 0 = failure).

This form assumes you want determined < allowable. (i.e.allowable is a maxAllowable). It could be generalized to handle minAllowable case also.

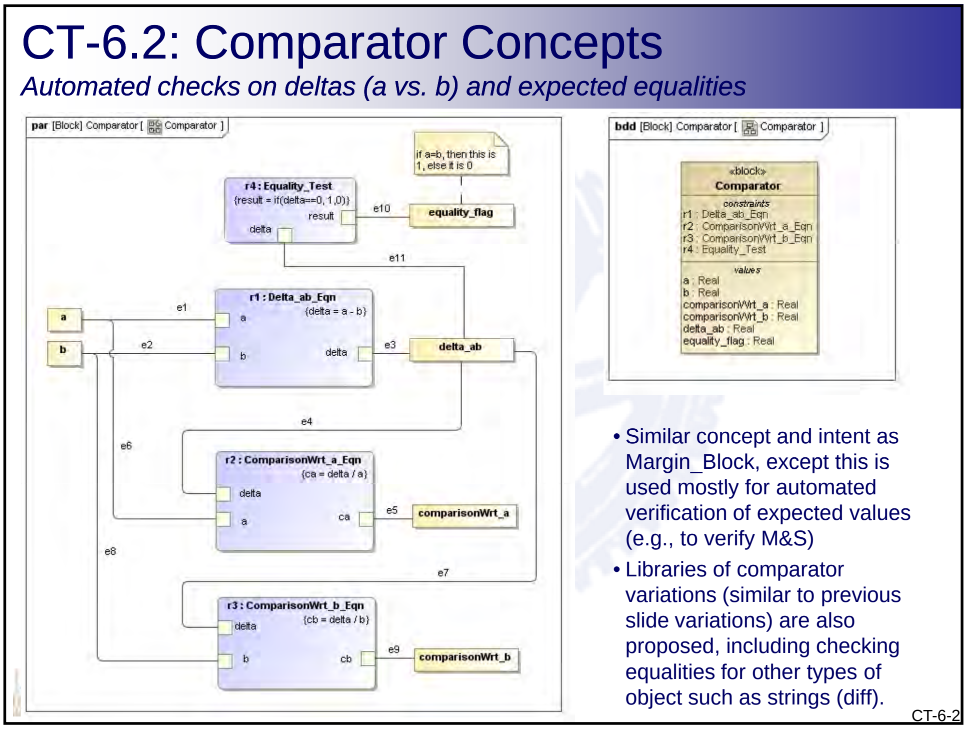
See also:
http://en.wikipedia.org/wiki/Factor_of_safety

if margin <0,
then the verdict=0 (fail)
otherwise the
verdict=1 (pass)

- Common concept used in aerospace industry, etc., for quantifying requirements & objectives (e.g., in stress analysis)
- Analogous to fundamental elements in electrical circuits (resistors, diodes, ...)
- Useful for non-physics contexts, too (e.g., target cash-on-hand in a business)

- *Suggested next steps:* Create reusable SysML libraries of such concepts to kick-start broader usage for V&V.
- *Other variations:* max vs. min allowables, ranges, tolerances, multiple levels of acceptability (not just pass/fail), expect delta trend vs. baseline ...

CT-6-1



CT-6.2: Comparator Concepts

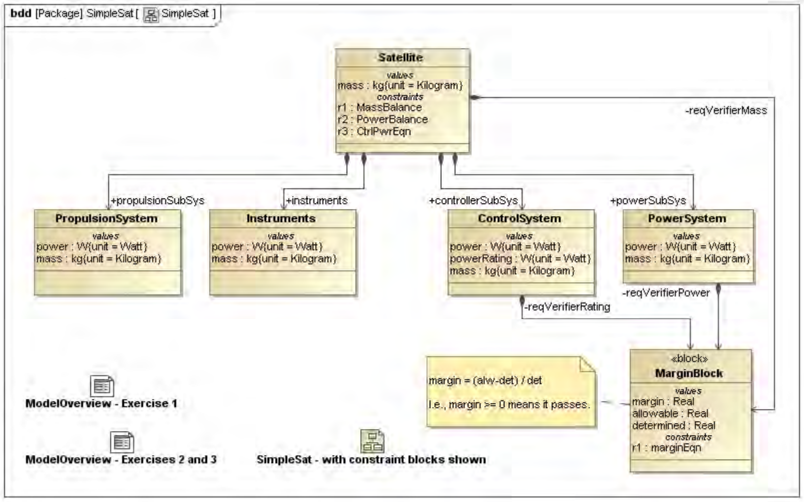
Automated checks on deltas (a vs. b) and expected equalities

- Similar concept and intent as Margin_Block, except this is used mostly for automated verification of expected values (e.g., to verify M&S)
- Libraries of comparator variations (similar to previous slide variations) are also proposed, including checking equalities for other types of object such as strings (diff).

CT-6-2

CT-7.1: Automated requirements verification

Example: SimpleSat Parametrics Tutorial (slide 1 of 4)

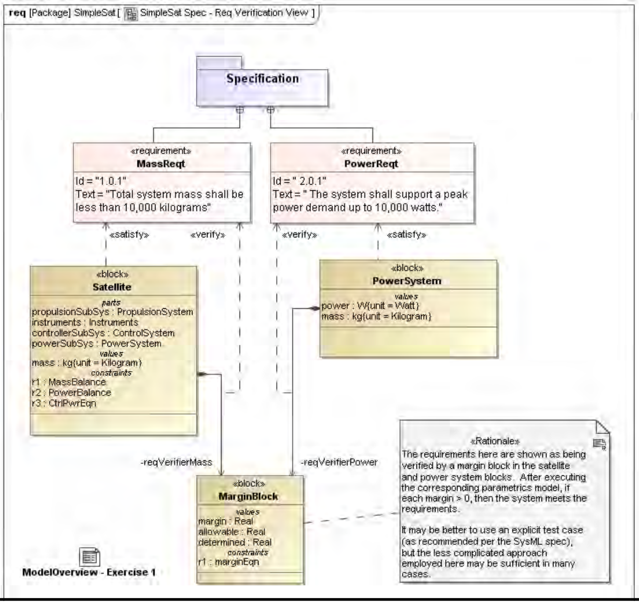


Georgia Institute of Technology

CT-7-1

CT-7.1: SimpleSat Parametrics Tutorial

req diagram showing requirements verification pattern


req [Package] SimpleSat [SimpleSat Spec - Req Verification View]
Specification
«requirement»
MassReqt
Id = "1.0.1"
Text = "Total system mass shall be less than 10,000 kilograms"
«requirement»
PowerReqt
Id = " 2.0.1"
Text = " The system shall support a peak power demand up to 10,000 watts."
«satisfy»
«verify»
«verify»
«satisfy»
«block»
Satellite
parts
propulsionSubSys : PropulsionSystem
instruments : Instruments
controllerSubSys : ControlSystem
powerSubSys : PowerSystem
values
mass : kg{unit = Kilogram}
constraints
r1 : MassBalance
r2 : PowerBalance
r3 : CtrlPwrEqn
«block»
PowerSystem
values
power : W{unit = Watt}
mass : kg{unit = Kilogram}
-reqVerifierMass
-reqVerifierPower
«block»
MarginBlock
values
margin : Real
allowable : Real
determined : Real
constraints
r1 : marginEqn
ModelOverview - Exercise 1

«Rationale»
The requirements here are shown as being verified by a margin block in the satellite and power system blocks. After executing the corresponding parametrics model, if each margin > 0, then the system meets the requirements.

It may be better to use an explicit test case (as recommended per the SysML spec), but the less complicated approach employed here may be sufficient in many cases.

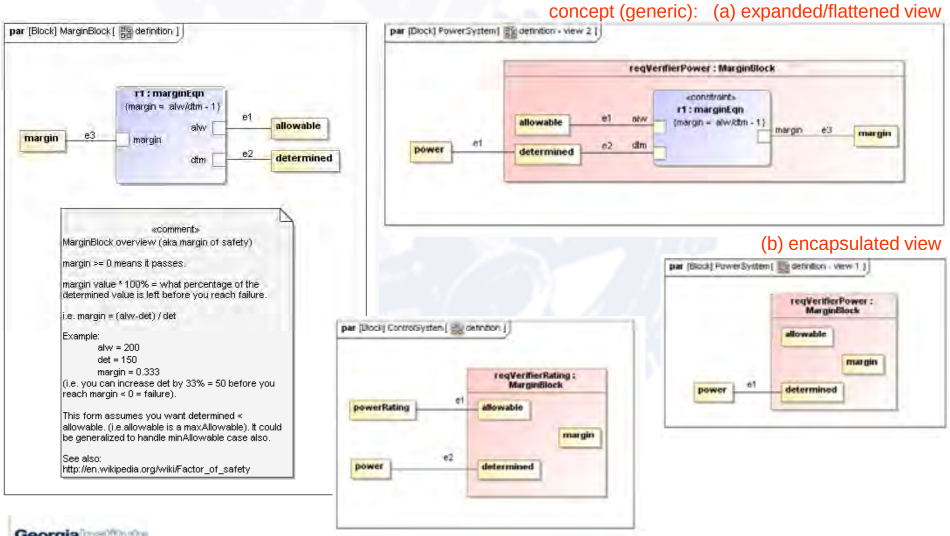
Georgia Institute of Technology

CT-7-2

CT-7.1: SimpleSat Parametrics Tutorial

par structure of building blocks and subsystems

concept (generic): (a) expanded/flattened view



par [Block] MarginBlock [definition]
r1 : marginEqn
{margin = alw/dtm - 1}
margin
e3
margin
alw
dtm
e1
e2
allowable
determined
par [Block] PowerSystem [definition - view 2]
reqVerifierPower : MarginBlock
«constraint»
r1 : marginEqn
{margin = alw/dtm - 1}
power
e1
allowable
determined
e1
e2
alw
dtm
margin
e3
margin
par [Block] ControlSystem [definition]
reqVerifierRating : MarginBlock
powerRating
e1
allowable
margin
power
e2
determined
par [Block] PowerSystem [definition - view 1]
reqVerifierPower : MarginBlock
allowable
margin
power
e1
determined

«comment»
MarginBlock overview (aka margin of safety)

margin >= 0 means it passes.

margin value * 100% = what percentage of the determined value is left before you reach failure.

i.e. margin = (alw-det) / det

Example:
alw = 200
det = 150
margin = 0.333
(i.e. you can increase det by 33% = 50 before you reach margin < 0 = failure).

This form assumes you want determined < allowable. (i.e.allowable is a maxAllowable). It could be generalized to handle minAllowable case also.

See also:
http://en.wikipedia.org/wiki/Factor_of_safety

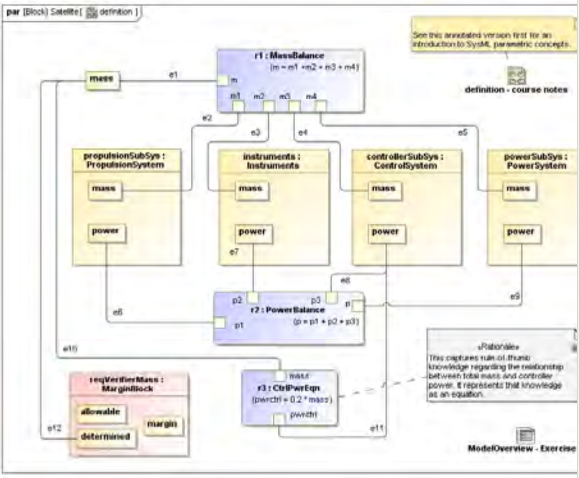
(b) encapsulated view

Georgia Institute of Technology

CT-7-3

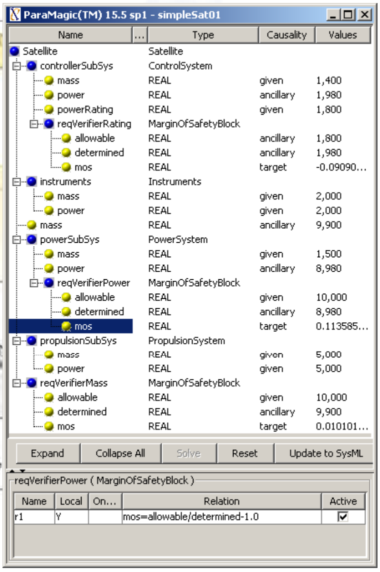
CT-7.1: SimpleSat Tutorial

SysML par view and ParaMagic tool for execution



See this annotated version first for an introduction to SysML parametric concepts.

«Rationale»
This captures rule-of-thumb knowledge regarding the relationship between total mass and controller power. It represents that knowledge as an equation.



Name	...	Type	Causality	Values
Satellite		Satellite		
controllerSubSys		ControlSystem		
mass		REAL	given	1,400
power		REAL	ancillary	1,980
powerRating		REAL	given	1,800
reqVerifierRating		MarginOfSafetyBlock		
allowable		REAL	ancillary	1,800
determined		REAL	ancillary	1,980
mos		REAL	target	-0.09090...
instruments		Instruments		
mass		REAL	given	2,000
power		REAL	given	2,000
mass		REAL	ancillary	9,900
powerSubSys		PowerSystem		
mass		REAL	given	1,500
power		REAL	ancillary	8,980
reqVerifierPower		MarginOfSafetyBlock		
allowable		REAL	given	10,000
determined		REAL	ancillary	8,980
mos		REAL	target	0.113585...
propulsionSubSys		PropulsionSystem		
mass		REAL	given	5,000
power		REAL	given	5,000
reqVerifierMass		MarginOfSafetyBlock		
allowable		REAL	given	10,000
determined		REAL	ancillary	9,900
mos		REAL	target	0.010101...

reqVerifierPower (MarginOfSafetyBlock)

Name	Local	On...	Relation	Active
r1	Y		mos=allowable/determined-1.0	☑

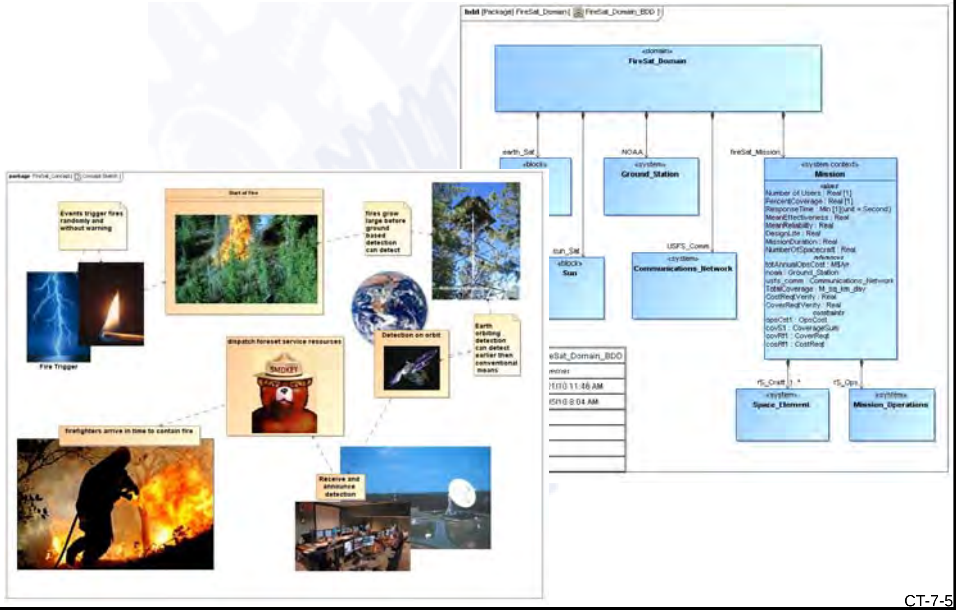
"Object-Oriented Spreadsheet"
plus more ...

Georgia Institute of Technology

CT-7-4

CT-7.2: Requirements Verification in FireSat

Sources: INCOSE SSWG and InterCAX LLC; Georgia Tech ASE 6006 NGDMC

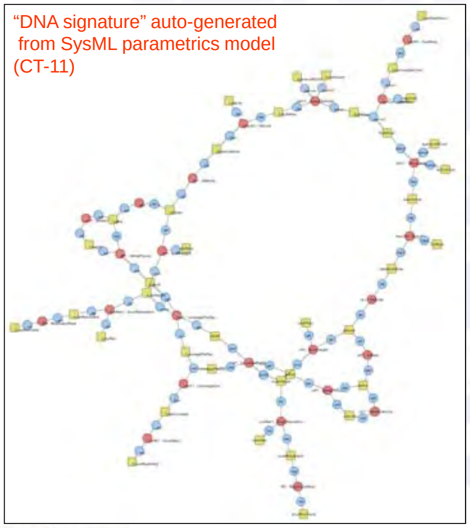

bdd [Package] FireSat_Domain [FireSat_Domain_BDD]
«domain»
FireSat_Domain
earth_Sat
«block»
NOAA
«system»
Ground_Station
fireSat_Mission
«system context»
Mission
values
Number of Users : Real [1]
PercentCoverage : Real [1]
ResponseTime : Min [1]{unit = Second}
MeanEffectiveness : Real
MeanReliability : Real
DesignLife : Real
MissionDuration : Real
NumberOfSpacecraft : Real
references
totAnnualOpsCost : M$/yr
noaa : Ground_Station
usfs_comm : Communications_Network
TotalCoverage : M_sq_km_day
CostReqtVerify : Real
CoverReqtVerify : Real
constraints
opsCst1 : OpsCost
covS1 : CoverageSum
covRt1 : CoverReqt
cosRt1 : CostReqt
sun_Sat
«block»
Sun
USFS_Comm
«system»
Communications_Network
fS_Craft 1..*
fS_Ops
«system»
Space_Element
«system»
Mission_Operations
package FireSat_Concept [Concept Sketch]
Events trigger fires randomly and without warning
Fire Trigger
Start of Fire
fires grow large before ground based detection can detect
Earth orbiting detection can detect earlier then conventional means
Detection on orbit
dispatch foreset service resourses
SMOKEY
firefighters arrive in time to contain fire
Receive and announce detection

CT-7-5

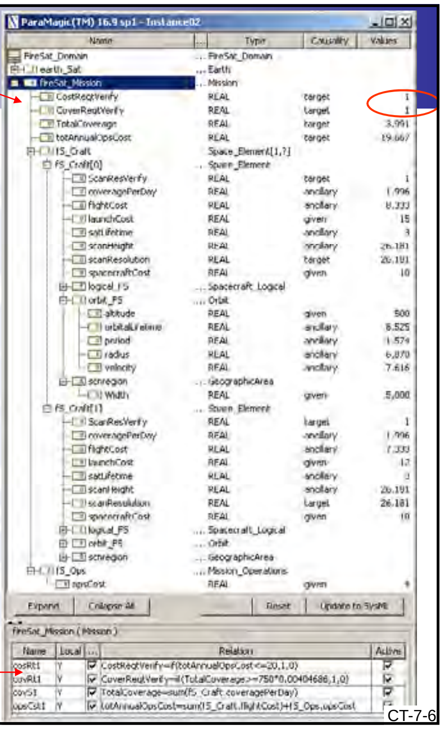
CT-7.2: Req. Verification

in FireSat SysML model

(including operational costs, etc.)



"DNA signature" auto-generated from SysML parametrics model (CT-11)



Name	...	Type	Causality	Values
FireSat_Domain	...	FireSat_Domain		
earth_Sat	...	Earth		
fireSat_Mission	...	Mission		
CostReqtVerify		REAL	target	1
CoverReqtVerify		REAL	target	1
TotalCoverage		REAL	target	3.991
totAnnualOpsCost		REAL	target	19.667
fS_Craft		Space_Element[1,?]		
fS_Craft[0]	...	Space_Element		
ScanResVerify		REAL	target	1
coveragePerDay		REAL	ancillary	1.996
flightCost		REAL	ancillary	8.333
launchCost		REAL	given	15
satLifetime		REAL	ancillary	3
scanHeight		REAL	ancillary	26.181
scanResolution		REAL	target	26.181
spacecraftCost		REAL	given	10
logical_FS	...	Spacecraft_Logical		
orbit_FS	...	Orbit		
altitude		REAL	given	500
orbitalLifetime		REAL	ancillary	6.525
period		REAL	ancillary	1.574
radius		REAL	ancillary	6,870
velocity		REAL	ancillary	7.616
scnregion	...	GeographicArea		
Width		REAL	given	5,000
fS_Craft[1]	...	Space_Element		
ScanResVerify		REAL	target	1
coveragePerDay		REAL	ancillary	1.996
flightCost		REAL	ancillary	7.333
launchCost		REAL	given	12
satLifetime		REAL	ancillary	3
scanHeight		REAL	ancillary	26.191
scanResolution		REAL	target	26.181
spacecraftCost		REAL	given	10
logical_FS	...	Spacecraft_Logical		
orbit_FS	...	Orbit		
scnregion	...	GeographicArea		
fS_Ops	...	Mission_Operations		
opsCost		REAL	given	4

fireSat_Mission (Mission)

Name	Local	...	Relation	Active
cosRt1	Y	☑	CostReqtVerify=if(totAnnualOpsCost<=20,1,0)	☑
covRt1	Y	☑	CoverReqtVerify=if(TotalCoverage>=750*0.00404686,1,0)	☑
covS1	Y	☑	TotalCoverage=sum(fS_Craft.coveragePerDay)	☑
opsCst1	Y	☑	totAnnualOpsCost=sum(fS_Craft.flightCost)+fS_Ops.opsCost	☐

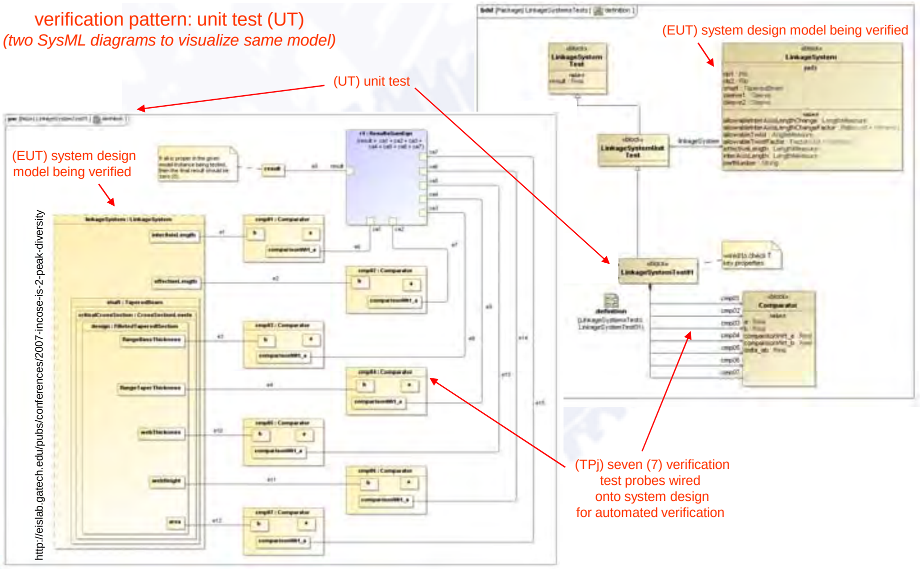
Georgia Institute of Technology

Model source: Dirk.Zwemer@InterCAX.com

CT-7-6

CT-8.1a: Unit Test/Verification Pattern

Verifying SysML model: Linkage Systems



verification pattern: unit test (UT)

(two SysML diagrams to visualize same model)

(UT) unit test

(EUT) system design model being verified

(EUT) system design model being verified

(TPj) seven (7) verification test probes wired onto system design for automated verification

Georgia Institute of Technology

EUT = entity-under-test

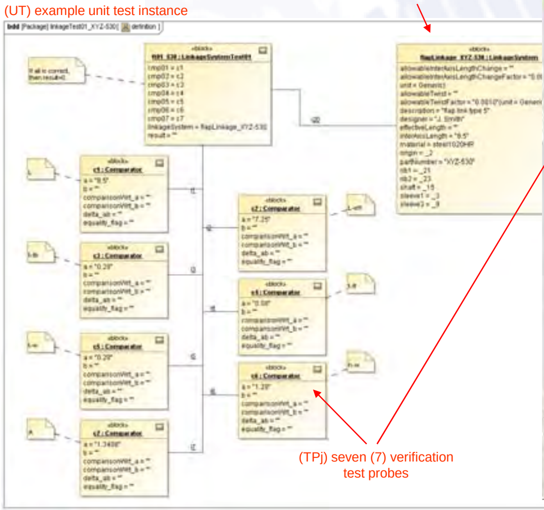
CT-8-1

CT-8.1a: Unit Test/Verification Pattern

Linkage Systems unit test: sample instances & execution

(UT) example unit test instance

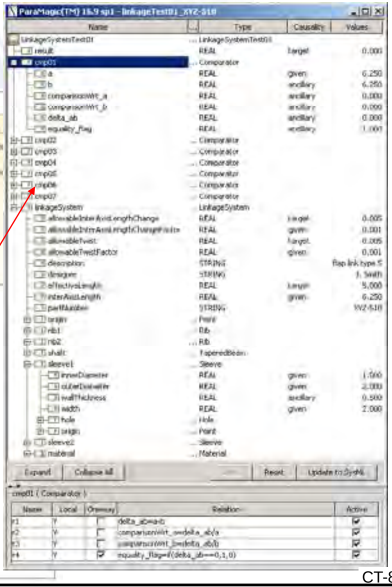
(EUT) system model instance being verified



bdd [Package] linkageTest01_XYZ-530[definition]
If all is correct, then result=0.
tst01_530 : LinkageSystemTest#1
flapLinkage_XYZ-530 : LinkageSystem
c1 : Comparator
c2 : Comparator
c3 : Comparator
c4 : Comparator
c5 : Comparator
c6 : Comparator
c7 : Comparator

(TPj) seven (7) verification test probes

automated execution in ParaMagic



Name	Type	Causality	Values
LinkageSystemTest01	LinkageSystemTest01		
result	REAL	target	0.000
cmp01	Comparator		
a	REAL	given	6.250
b	REAL	ancillary	6.250
comparisonWrt_a	REAL	ancillary	0.000
comparisonWrt_b	REAL	ancillary	0.000
delta_ab	REAL	ancillary	0.000
equality_flag	REAL	ancillary	1.000
cmp02	Comparator		
cmp03	Comparator		
cmp04	Comparator		
cmp05	Comparator		
cmp06	Comparator		
cmp07	Comparator		
linkageSystem	LinkageSystem		
allowableInterAxisLengthChange	REAL	target	0.005
allowableInterAxisLengthChangeFactor	REAL	given	0.001
allowableTwist	REAL	target	0.005
allowableTwistFactor	REAL	given	0.001
description	STRING		flap link type 5
designer	STRING		J. Smith
effectiveLength	REAL	target	5.000
interAxisLength	REAL	given	6.250
partNumber	STRING		XYZ-510
origin	Point		
rib1	Rib		
rib2	Rib		
shaft	TaperedBeam		
sleeve1	Sleeve		
innerDiameter	REAL	given	1.000
outerDiameter	REAL	given	2.000
wallThickness	REAL	ancillary	0.500
width	REAL	given	2.000
hole	Hole		
origin	Point		
sleeve2	Sleeve		
material	Material		

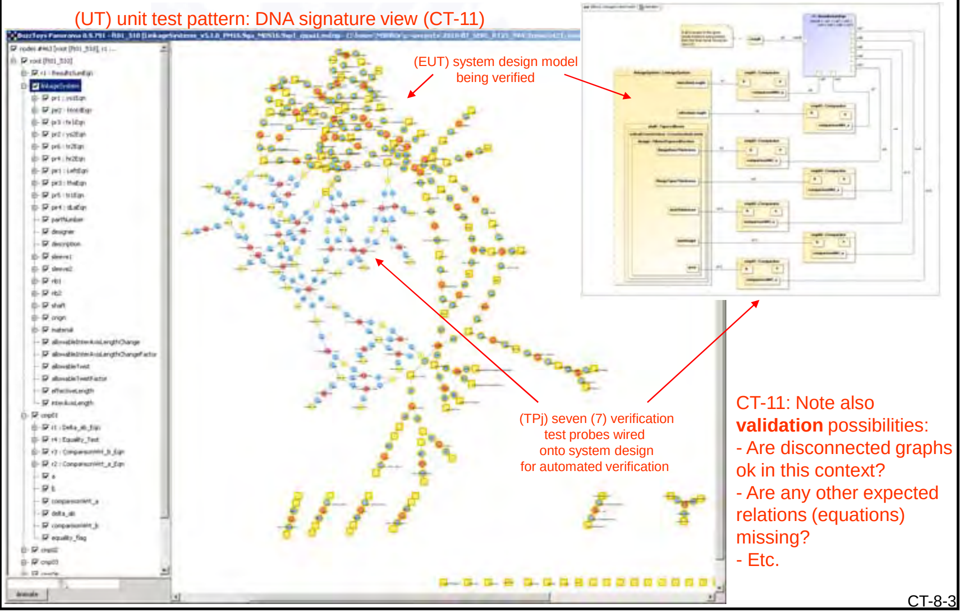
cmp01 (Comparator)

Name	Local	Oneway	Relation	Active
r1	Y	☐	delta_ab=a-b	☑
r2	Y	☐	comparisonWrt_a=delta_ab/a	☑
r3	Y	☐	comparisonWrt_b=delta_ab/b	☑
r4	Y	☑	equality_flag=if(delta_ab==0,1,0)	☑

CT-8-2

CT-8.1a: Unit Test/Verification Pattern

Verifying SysML model: Linkage Systems



(UT) unit test pattern: DNA signature view (CT-11)

(EUT) system design model being verified

(TPj) seven (7) verification test probes wired onto system design for automated verification

CT-11: Note also **validation** possibilities:
- Are disconnected graphs ok in this context?
- Are any other expected relations (equations) missing?
- Etc.

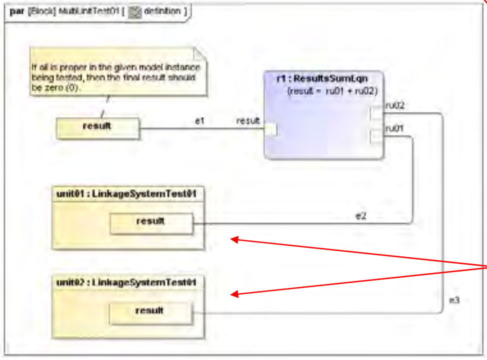
CT-8-3

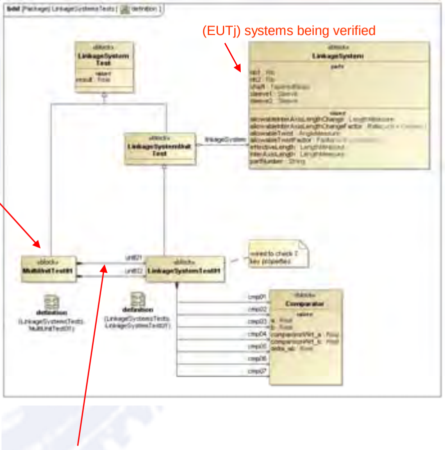
CT-9.1: Multi-Unit Test/Verification Pattern

Chaining several unit tests to verify SysML model Linkage Systems

verification pattern: multi-unit test (MUT)
(two SysML diagrams to visualize same model)

(MUT) multi-unit test


par [Block] MultiUnitTest01 [definition]
If all is proper in the given model instance being tested, then the final result should be zero (0).
result
e1
result
r1 : ResultsSumEqn
{result = ru01 + ru02}
ru02
ru01
unit01 : LinkageSystemTest01
result
e2
unit02 : LinkageSystemTest01
result
e3



(EUTj) systems being verified

(UTj) multiple uses of same unit test (CT-8.1)
(two in this case)

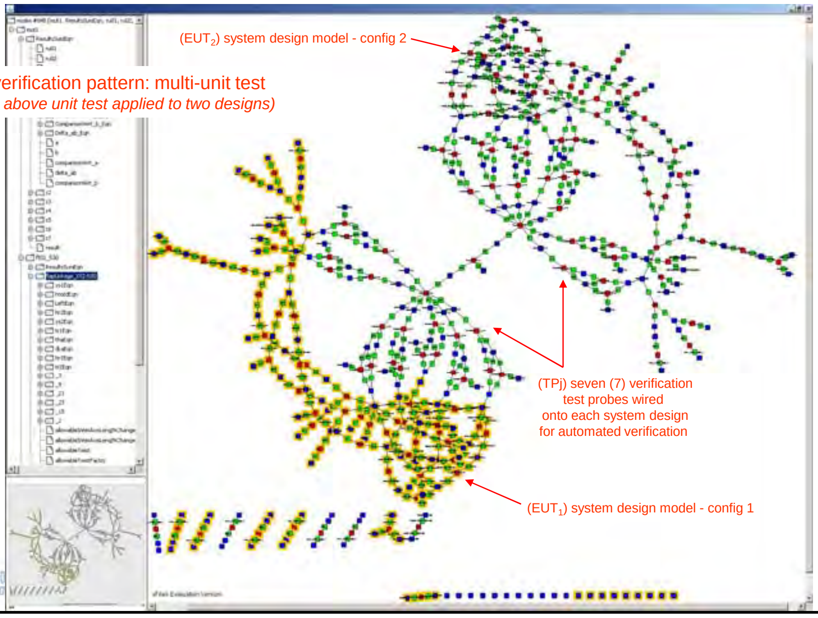
Note: In this case each UTj is the same type of unit test, but in general a single MUT can support different types of UTj

EUT = entity-under-test

CT-9-1

CT-9.1: Multi-Unit Test (Verification Suite)

CT-11: "DNA signature" auto-generated from SysML parametrics model



(EUT$_2$) system design model - config 2

(MUT) verification pattern: multi-unit test

(rolling up above unit test applied to two designs)

(TPj) seven (7) verification test probes wired onto each system design for automated verification

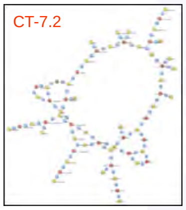
(EUT$_1$) system design model - config 1

CT-9-2

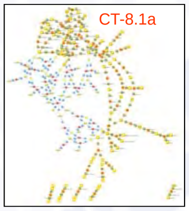
CT-11: "DNA signatures"

User interaction with models for intuitive visual inspection to aid model comprehension, V&V, debugging, etc.

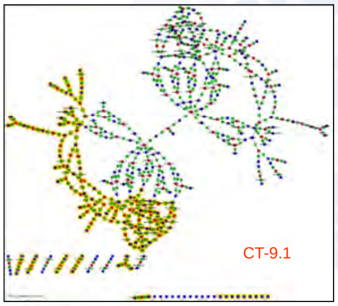
Selected examples from other CT-n sections



CT-7.2

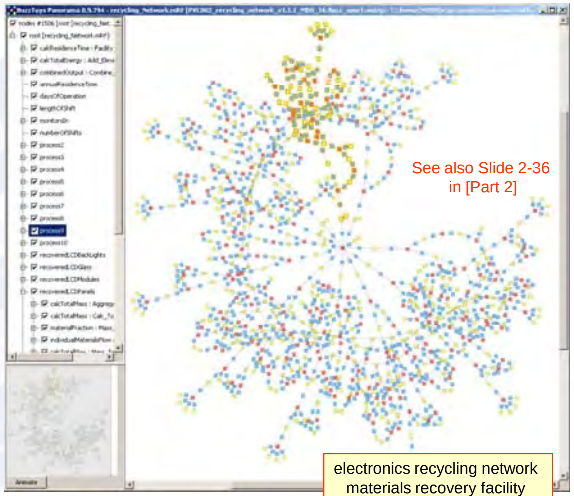


CT-8.1a



CT-9.1

CT-11.3 - pinwheel roll-up pattern

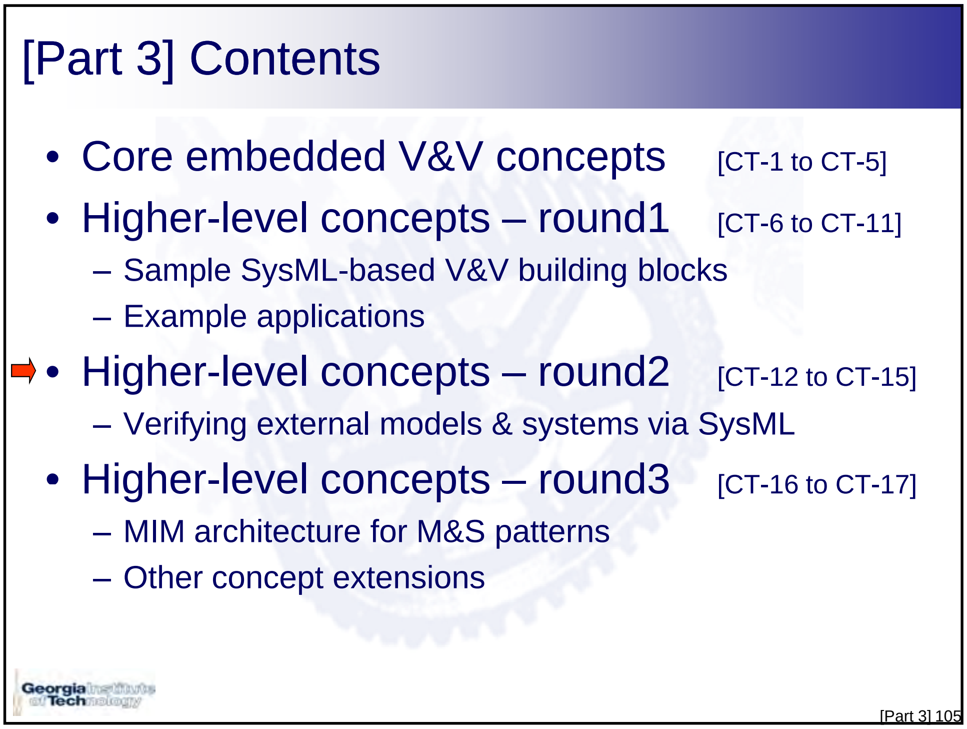


See also Slide 2-36 in [Part 2]

electronics recycling network materials recovery facility with 11 processes

DNA signatures auto-generated from SysML parametrics structure

CT-11-1

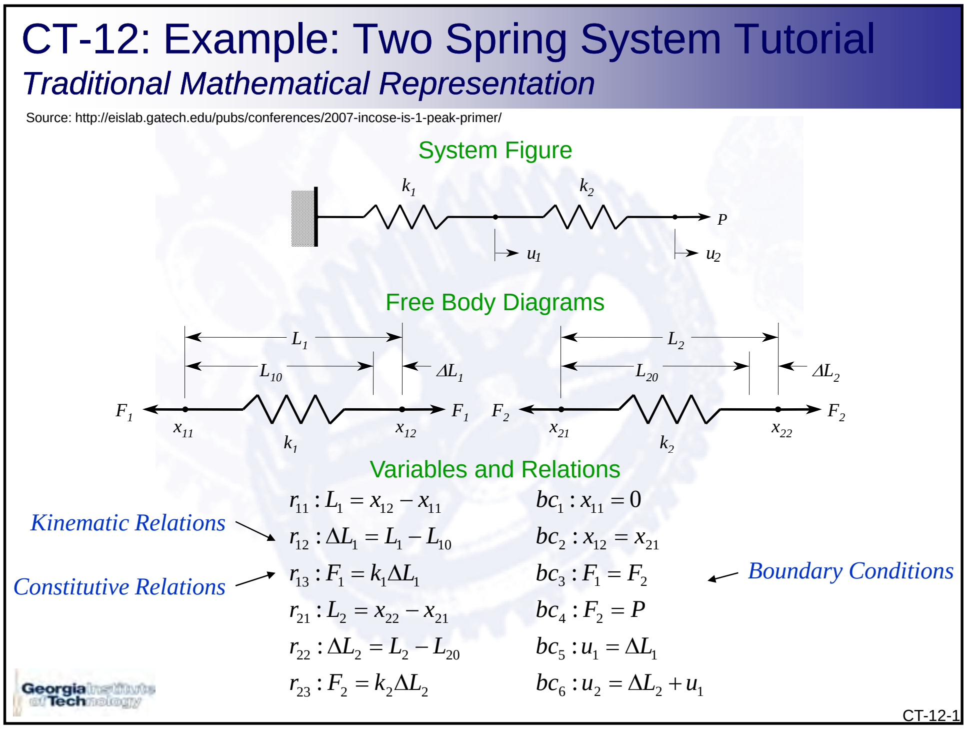


[Part 3] Contents

- Core embedded V&V concepts [CT-1 to CT-5]
- Higher-level concepts – round1 [CT-6 to CT-11]
  - Sample SysML-based V&V building blocks
  - Example applications
- ➡ Higher-level concepts – round2 [CT-12 to CT-15]
  - Verifying external models & systems via SysML
- Higher-level concepts – round3 [CT-16 to CT-17]
  - MIM architecture for M&S patterns
  - Other concept extensions

Georgia Institute of Technology

[Part 3] 105



CT-12: Example: Two Spring System Tutorial

Traditional Mathematical Representation

Source: http://eislab.gatech.edu/pubs/conferences/2007-incose-is-1-peak-primer/

System Figure

Free Body Diagrams

Variables and Relations

Kinematic Relations

Constitutive Relations

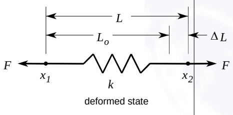
Boundary Conditions

$r_{11} : L_1 = x_{12} - x_{11}$	$bc_1 : x_{11} = 0$
$r_{12} : \Delta L_1 = L_1 - L_{10}$	$bc_2 : x_{12} = x_{21}$
$r_{13} : F_1 = k_1 \Delta L_1$	$bc_3 : F_1 = F_2$
$r_{21} : L_2 = x_{22} - x_{21}$	$bc_4 : F_2 = P$
$r_{22} : \Delta L_2 = L_2 - L_{20}$	$bc_5 : u_1 = \Delta L_1$
$r_{23} : F_2 = k_2 \Delta L_2$	$bc_6 : u_2 = \Delta L_2 + u_1$

Georgia Institute of Technology

CT-12-1

Spring System Example

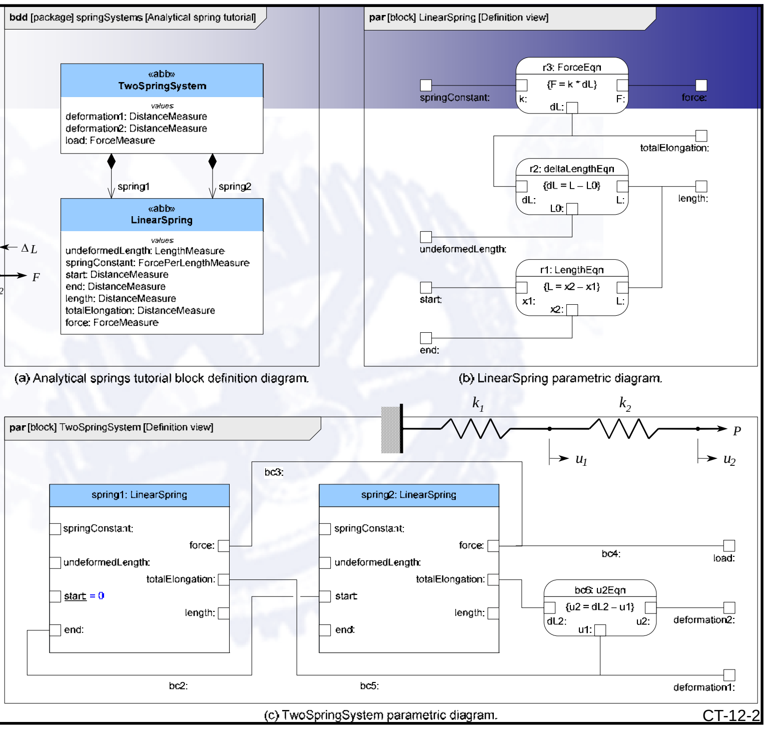


$r_1 : L = x_2 - x_1$

$r_2 : \Delta L = L - L_0$

$r_3 : F = k\Delta L$

SysML Diagrams



(a) Analytical springs tutorial block definition diagram.

(b) LinearSpring parametric diagram.

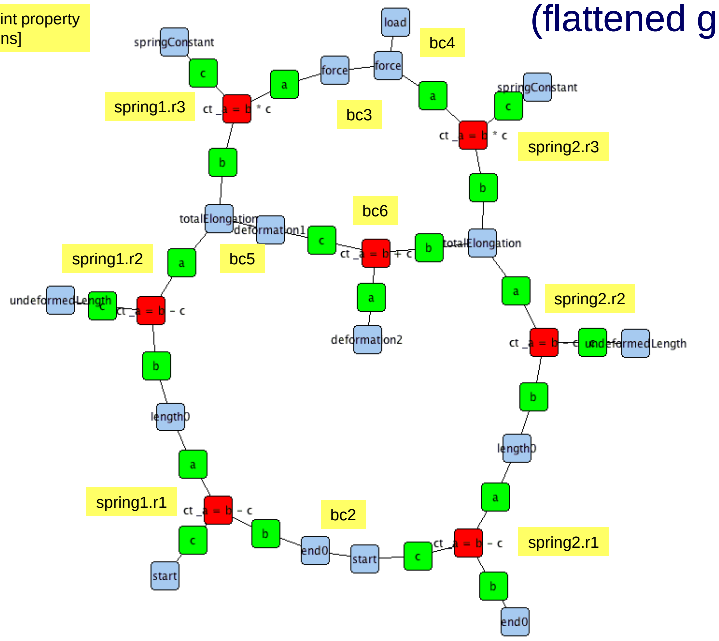
(c) TwoSpringSystem parametric diagram.

CT-12-2

Spring System: DNA signature (CT-11)

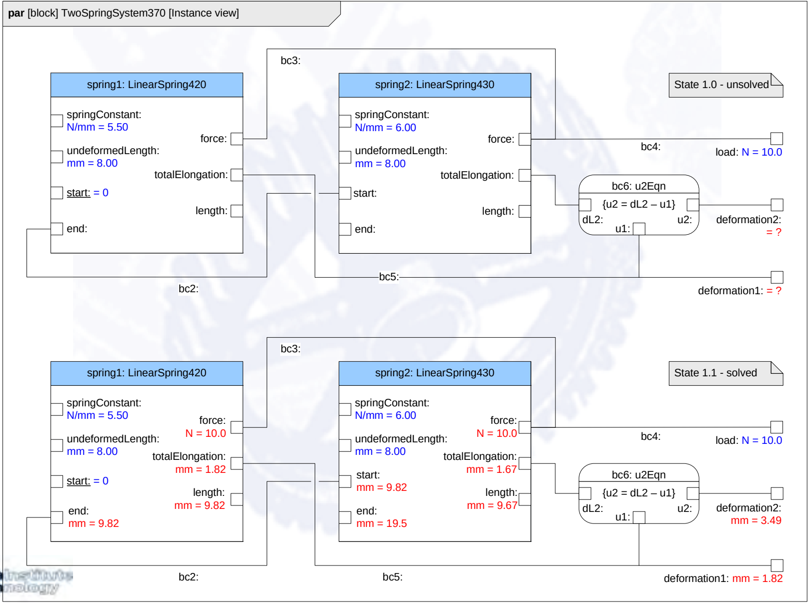
(flattened graph)

[SysML constraint property name annotations]



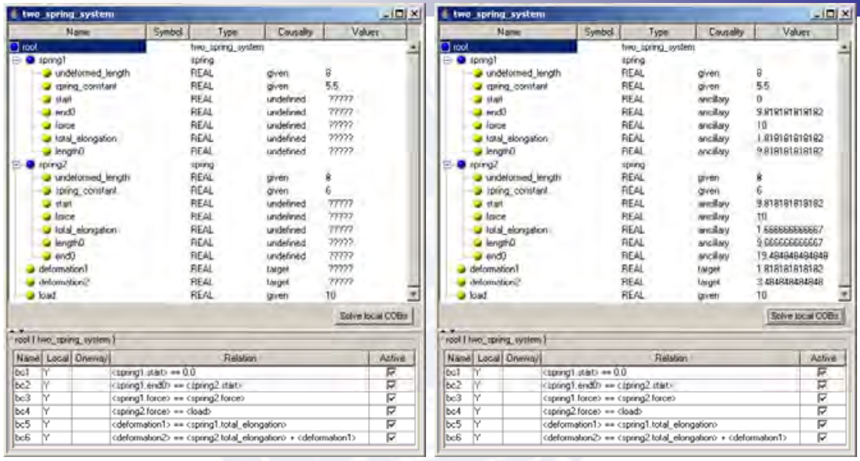
CT-12-3

TwoSpringSystem parametric diagram – sample instance



CT-12-4

Example instance: two_spring_system



Name	Symbol	Type	Causality	Values
root		two_spring_system		
spring1		spring		
undeformed_length		REAL	given	8
spring_constant		REAL	given	5.5
start		REAL	undefined	?????
end0		REAL	undefined	?????
force		REAL	undefined	?????
total_elongation		REAL	undefined	?????
length0		REAL	undefined	?????
spring2		spring		
undeformed_length		REAL	given	8
spring_constant		REAL	given	6
start		REAL	undefined	?????
force		REAL	undefined	?????
total_elongation		REAL	undefined	?????
length0		REAL	undefined	?????
end0		REAL	undefined	?????
deformation1		REAL	target	?????
deformation2		REAL	target	?????
load		REAL	given	10

Name	Local	Oneway	Relation	Active
bc1	Y		<spring1.start> == 0.0	☑
bc2	Y		<spring1.end0> == <spring2.start>	☑
bc3	Y		<spring1.force> == <spring2.force>	☑
bc4	Y		<spring2.force> == <load>	☑
bc5	Y		<deformation1> == <spring1.total_elongation>	☑
bc6	Y		<deformation2> == <spring2.total_elongation> + <deformation1>	☑

example 2, state 1.0 (unsolved)

Name	Symbol	Type	Causality	Values
root		two_spring_system		
spring1		spring		
undeformed_length		REAL	given	8
spring_constant		REAL	given	5.5
start		REAL	ancillary	0
end0		REAL	ancillary	9.81818181818182
force		REAL	ancillary	10
total_elongation		REAL	ancillary	1.81818181818182
length0		REAL	ancillary	9.81818181818182
spring2		spring		
undeformed_length		REAL	given	8
spring_constant		REAL	given	6
start		REAL	ancillary	9.81818181818182
force		REAL	ancillary	10
total_elongation		REAL	ancillary	1.66666666666667
length0		REAL	ancillary	9.66666666666667
end0		REAL	ancillary	19.4848484848484848
deformation1		REAL	target	1.81818181818182
deformation2		REAL	target	3.48484848484848
load		REAL	given	10

Name	Local	Oneway	Relation	Active
bc1	Y		<spring1.start> == 0.0	☑
bc2	Y		<spring1.end0> == <spring2.start>	☑
bc3	Y		<spring1.force> == <spring2.force>	☑
bc4	Y		<spring2.force> == <load>	☑
bc5	Y		<deformation1> == <spring1.total_elongation>	☑
bc6	Y		<deformation2> == <spring2.total_elongation> + <deformation1>	☑

example 2, state 1.1 (solved)

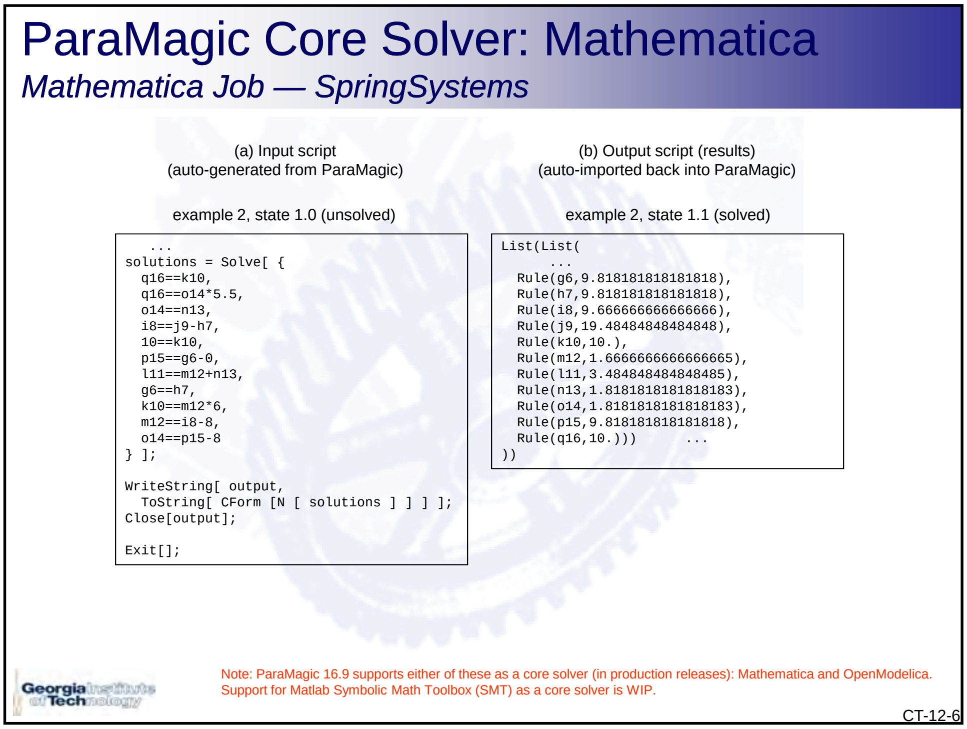
(b) Parametrics execution in *XaiTools / ParaMagic*

(a) Lexical COB instance as XML (CXI)

```
<linear_spring loid="_15">
  <undeformed_length causality="given">8.0</undeformed_length>
  <spring_constant causality="given">5.5</spring_constant>
</linear_spring>

<linear_spring loid="_25">
  <undeformed_length causality="given">8.0</undeformed_length>
  <spring_constant causality="given">6.0</spring_constant>
</linear_spring>
```

```
<two_spring_system loid="_3">
  <spring1 ref="_15"/>
  <spring2 ref="_25"/>
  <deformation1 causality="target"/>
  <deformation2 causality="target"/>
  <load causality="given">10.0</load>
</two_spring_system>
```

CT-12-5



ParaMagic Core Solver: Mathematica

Mathematica Job — SpringSystems

(a) Input script
(auto-generated from ParaMagic)

example 2, state 1.0 (unsolved)

```
   ...
solutions = Solve[ {
  q16==k10,
  q16==o14*5.5,
  o14==n13,
  i8==j9-h7,
  10==k10,
  p15==g6-0,
  l11==m12+n13,
  g6==h7,
  k10==m12*6,
  m12==i8-8,
  o14==p15-8
} ];

WriteString[ output,
  ToString[ CForm [N [ solutions ] ] ] ];
Close[output];

Exit[];
```

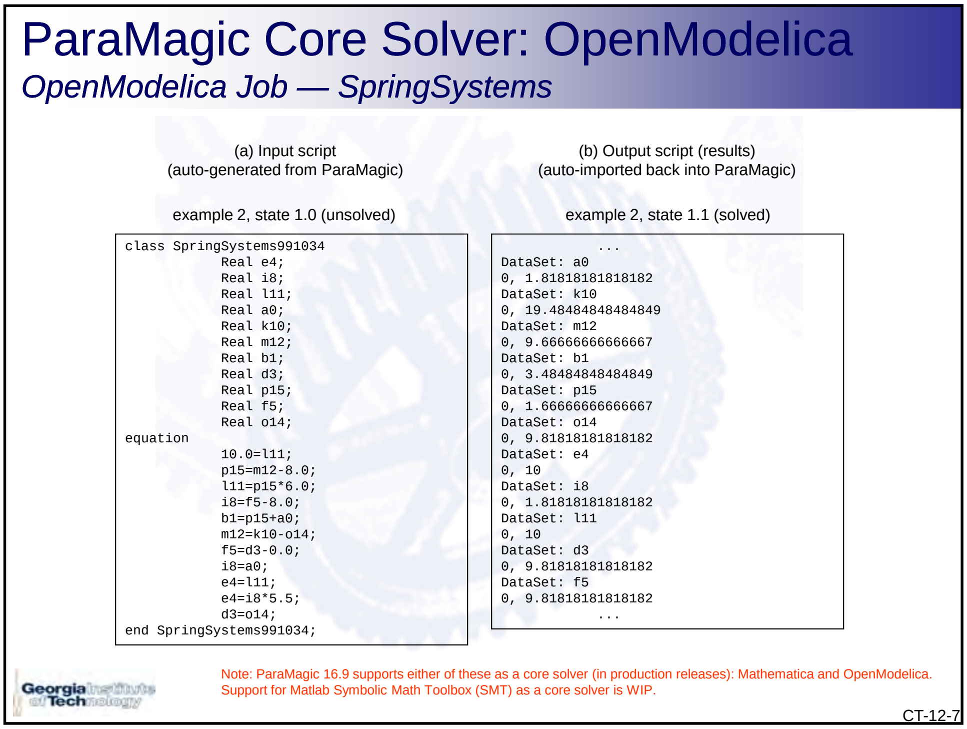
(b) Output script (results)
(auto-imported back into ParaMagic)

example 2, state 1.1 (solved)

```
List(List(
      ...
  Rule(g6,9.818181818181818),
  Rule(h7,9.818181818181818),
  Rule(i8,9.666666666666666),
  Rule(j9,19.48484848484848),
  Rule(k10,10.),
  Rule(m12,1.6666666666666665),
  Rule(l11,3.484848484848485),
  Rule(n13,1.8181818181818183),
  Rule(o14,1.8181818181818183),
  Rule(p15,9.818181818181818),
  Rule(q16,10.)))      ...
))
```

Note: ParaMagic 16.9 supports either of these as a core solver (in production releases): Mathematica and OpenModelica.
Support for Matlab Symbolic Math Toolbox (SMT) as a core solver is WIP.

CT-12-6



ParaMagic Core Solver: OpenModelica

OpenModelica Job — SpringSystems

(a) Input script
(auto-generated from ParaMagic)

example 2, state 1.0 (unsolved)

```
class SpringSystems991034
            Real e4;
            Real i8;
            Real l11;
            Real a0;
            Real k10;
            Real m12;
            Real b1;
            Real d3;
            Real p15;
            Real f5;
            Real o14;
equation
            10.0=l11;
            p15=m12-8.0;
            l11=p15*6.0;
            i8=f5-8.0;
            b1=p15+a0;
            m12=k10-o14;
            f5=d3-0.0;
            i8=a0;
            e4=l11;
            e4=i8*5.5;
            d3=o14;
end SpringSystems991034;
```

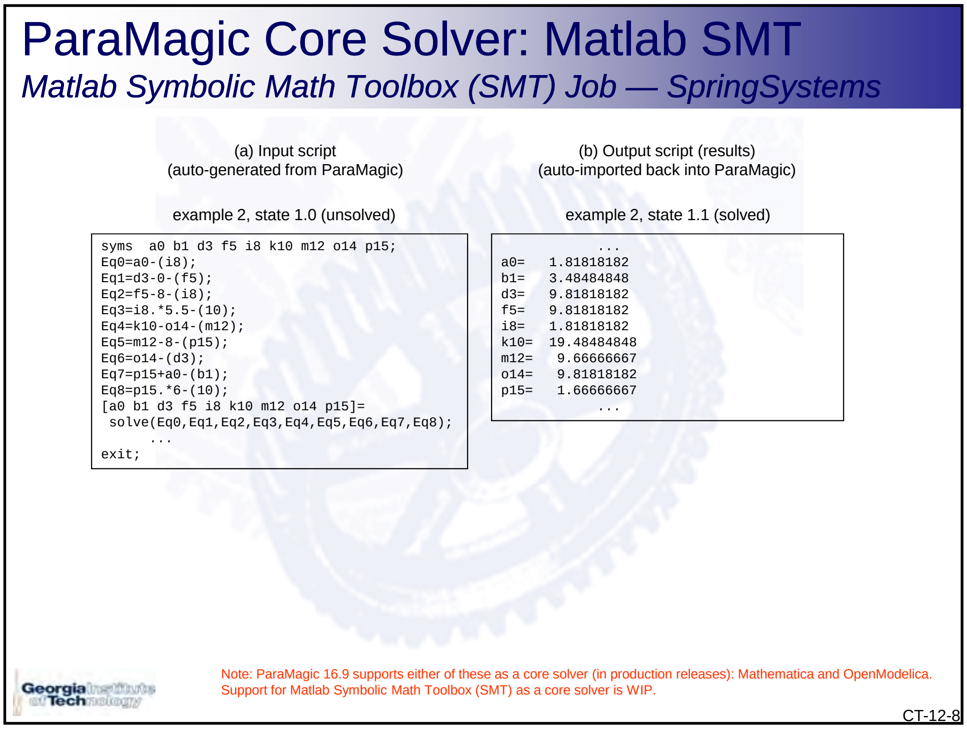
(b) Output script (results)
(auto-imported back into ParaMagic)

example 2, state 1.1 (solved)

```
             ...
DataSet: a0
0, 1.81818181818182
DataSet: k10
0, 19.48484848484849
DataSet: m12
0, 9.66666666666667
DataSet: b1
0, 3.48484848484849
DataSet: p15
0, 1.66666666666667
DataSet: o14
0, 9.81818181818182
DataSet: e4
0, 10
DataSet: i8
0, 1.81818181818182
DataSet: l11
0, 10
DataSet: d3
0, 9.81818181818182
DataSet: f5
0, 9.81818181818182
             ...
```

Note: ParaMagic 16.9 supports either of these as a core solver (in production releases): Mathematica and OpenModelica.
Support for Matlab Symbolic Math Toolbox (SMT) as a core solver is WIP.

CT-12-7



ParaMagic Core Solver: Matlab SMT

Matlab Symbolic Math Toolbox (SMT) Job — SpringSystems

(a) Input script
(auto-generated from ParaMagic)

example 2, state 1.0 (unsolved)

```
syms  a0 b1 d3 f5 i8 k10 m12 o14 p15;
Eq0=a0-(i8);
Eq1=d3-0-(f5);
Eq2=f5-8-(i8);
Eq3=i8.*5.5-(10);
Eq4=k10-o14-(m12);
Eq5=m12-8-(p15);
Eq6=o14-(d3);
Eq7=p15+a0-(b1);
Eq8=p15.*6-(10);
[a0 b1 d3 f5 i8 k10 m12 o14 p15]=
 solve(Eq0,Eq1,Eq2,Eq3,Eq4,Eq5,Eq6,Eq7,Eq8);
      ...
exit;
```

(b) Output script (results)
(auto-imported back into ParaMagic)

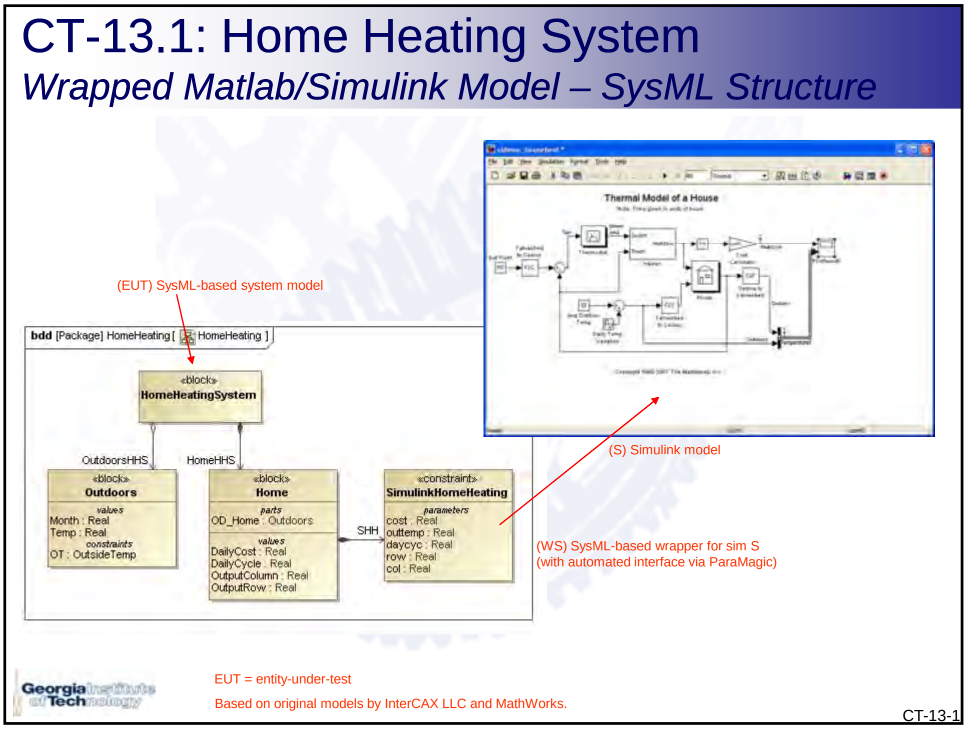
example 2, state 1.1 (solved)

```
              ...
a0=   1.81818182
b1=   3.48484848
d3=   9.81818182
f5=   9.81818182
i8=   1.81818182
k10=  19.48484848
m12=   9.66666667
o14=   9.81818182
p15=   1.66666667
              ...
```

Note: ParaMagic 16.9 supports either of these as a core solver (in production releases): Mathematica and OpenModelica. Support for Matlab Symbolic Math Toolbox (SMT) as a core solver is WIP.

Georgia Institute of Technology

CT-12-8



CT-13.1: Home Heating System

Wrapped Matlab/Simulink Model – SysML Structure

(EUT) SysML-based system model

(S) Simulink model

(WS) SysML-based wrapper for sim S
(with automated interface via ParaMagic)

EUT = entity-under-test

Based on original models by InterCAX LLC and MathWorks.

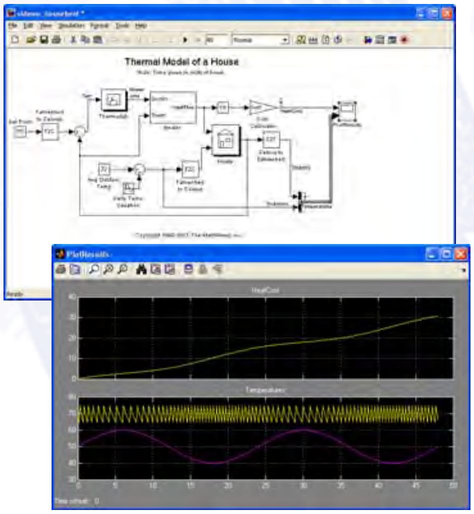
Georgia Institute of Technology

CT-13-1

CT-13.1: Home Heating System

Sample Matlab/Simulink Results

(S) SysML-wrapped system dynamics model
(home heating system in Matlab/Simulink)

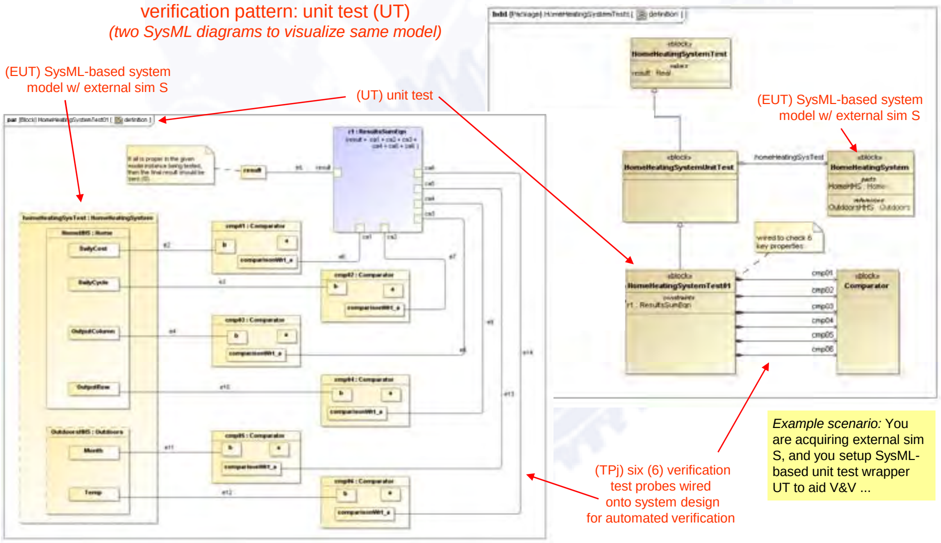


CT-13-2

CT-13.1/CT-8.1b: Home Heating System

Wrapped Matlab/Simulink Model – V&V Pattern

verification pattern: unit test (UT)
(two SysML diagrams to visualize same model)



(EUT) SysML-based system model w/ external sim S

(UT) unit test

(EUT) SysML-based system model w/ external sim S

(TPj) six (6) verification test probes wired onto system design for automated verification

Example scenario: You are acquiring external sim S, and you setup SysML-based unit test wrapper UT to aid V&V ...

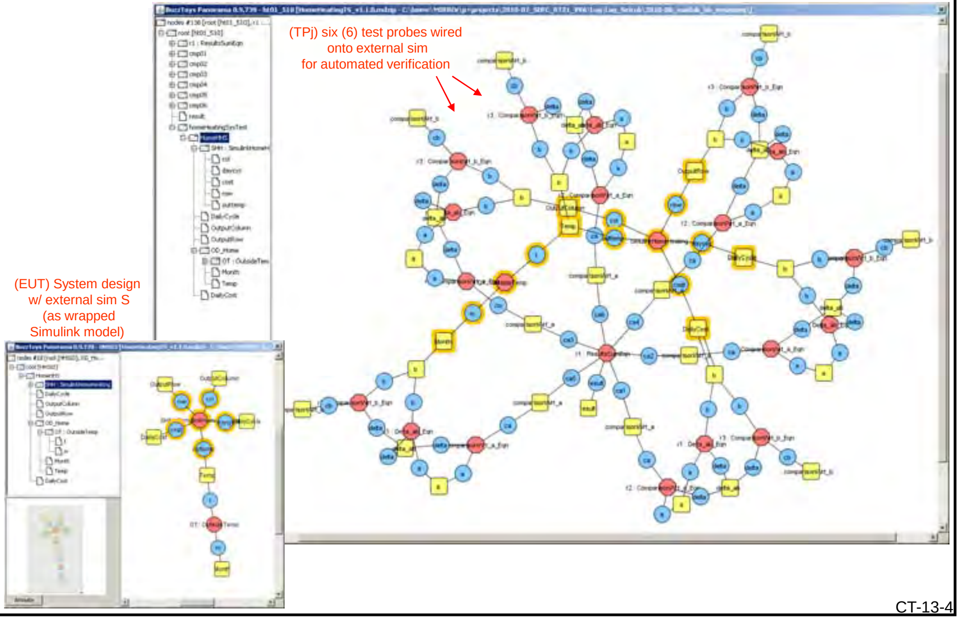
EUT = entity-under-test

SysML-based V&V added around original models by InterCAX LLC and MathWorks.

CT-13-3

CT-13.1: Sim Unit Test/V&V: Home Heating System

CT-11: DNA signature auto-generated from SysML model



(TPj) six (6) test probes wired onto external sim for automated verification

(EUT) System design w/ external sim S (as wrapped Simulink model)

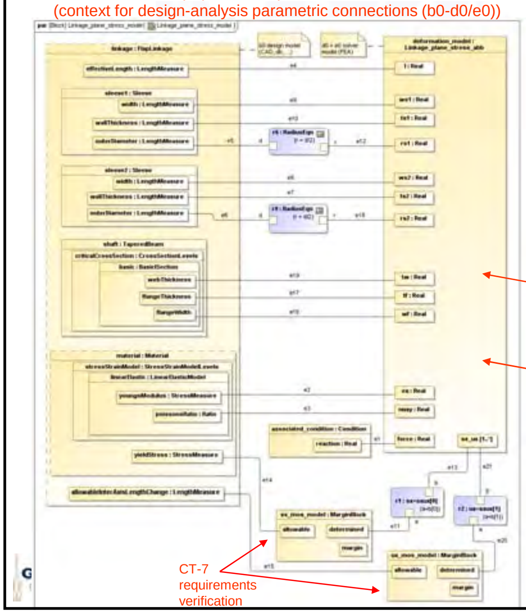
CT-13-4

CT-13.2: Wrapping Solver Models

FEA (Ansys) in MIM pattern context (CT-16)

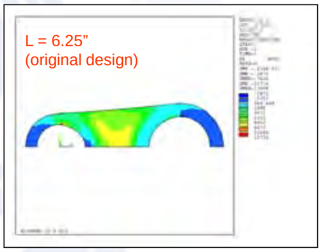
Can apply CT-8 unit test pattern here for automated V&V

(c0) context-specific model (context for design-analysis parametric connections (b0-d0/e0))

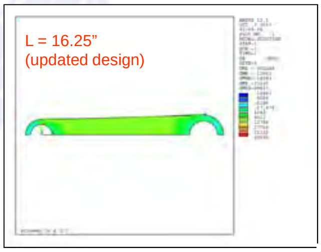


CT-7 requirements verification

(e0) SysML-wrapped FEA models (linkage systems in Ansys)



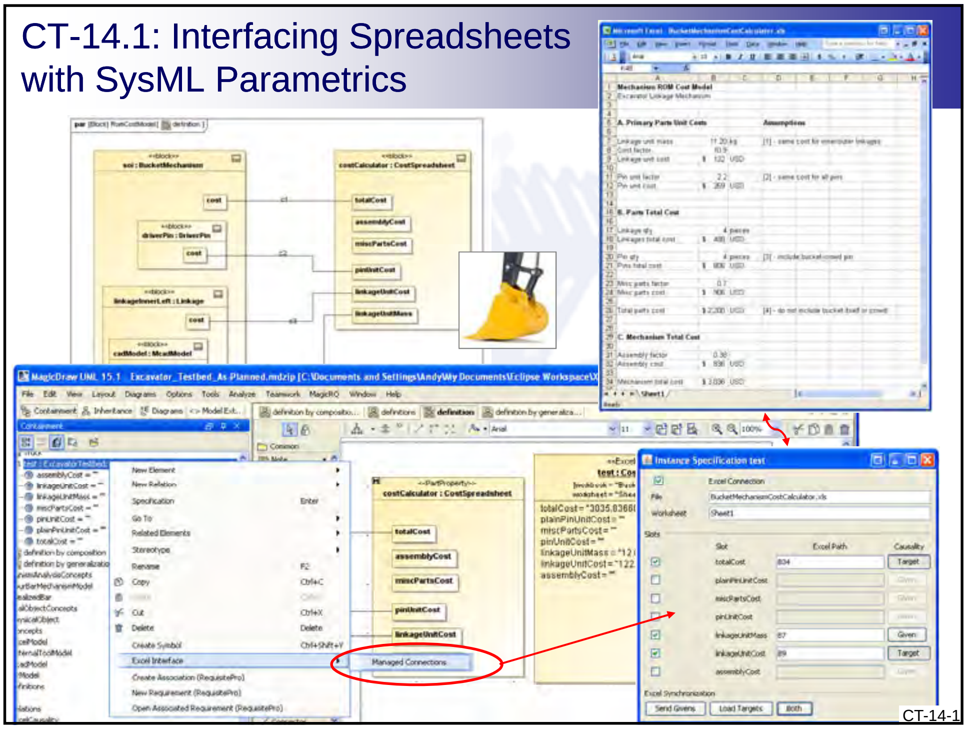
L = 6.25”
(original design)



L = 16.25”
(updated design)

CT-13-5

CT-14.1: Interfacing Spreadsheets with SysML Parametrics



CT-14-1

CT-14.2: Connecting a System Model to Domain Models (MCAD/ECAD) via SysML

Title: Composable Mission Framework for Rapid End-to-End Mission Design and Simulation

Principal Investigator: Dr. Manas Bajaj, InterCAX LLC

Phase 1: Jan–Jul, 2009 [NASA SBIR-08-1-S4.02-9130] — NASA SBIR project

Technical Abstract: The innovation proposed here is the Composable Mission Framework (CMF)—a model-based software framework that shall enable seamless continuity of mission design and simulation from early stage advanced studies to detailed mission design and development. The uniqueness of our approach lies in using an open standard for systems modeling and design (SysML) to wrap mission models including the mission development process thus providing a coherent map of mission knowledge. InterCAX's Composable Object technology provides the backend wrapping, model management, and simulation orchestration capabilities to the visual SysML-based mission model at the front end.

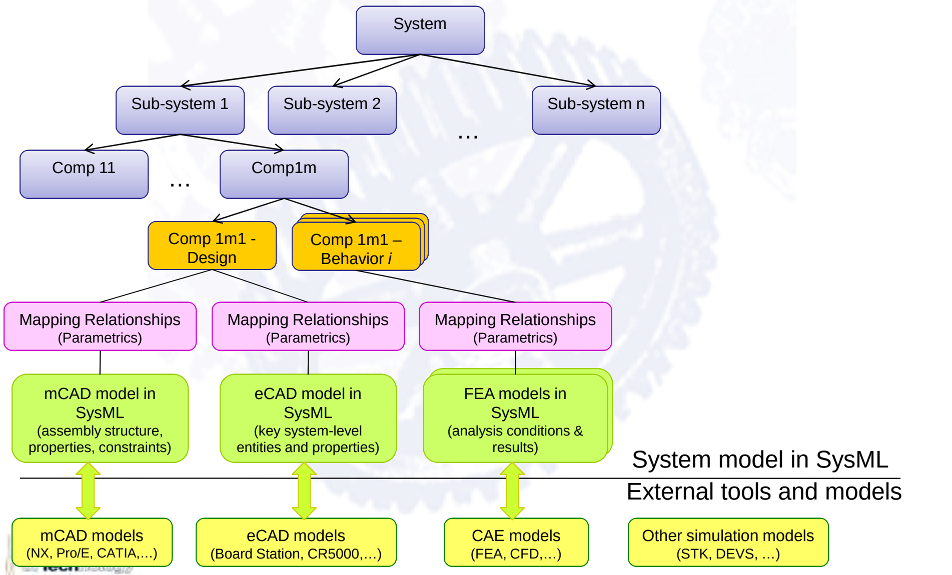
The Composable Object technology has already demonstrated the ability to power SysML-based models with math simulation capabilities for early design stages. ParaMagic is a commercially available tool being used by early adopters of SysML at JPL. The Composable Object technology has also demonstrated the ability to associate detailed design and simulation models such as those created in CAD and FEA tools. However, a big gap exists in the SysML-based world for conceptual system design and the detailed system design-based world. If the detailed system design and simulation models could be wrapped as SysML objects and the simulations and workflows orchestrated by the Composable Object technology, it will cover the entire gamut of complex system modeling and analysis world from trade studies and optimization to project scheduling.

The key objective of Phase 1 is to wrap both conceptual and detailed system design and simulation models as SysML objects which has not been done before, and to demonstrate continuity of mission concepts from simple to detailed implementation.

CT-14-2

CT-14.2

System Design & Analysis
Integrating and Executing Diverse Models



System

Sub-system 1

Sub-system 2

…

Sub-system n

Comp 11

…

Comp1m

Comp 1m1 - Design

Comp 1m1 – Behavior *i*

Mapping Relationships (Parametrics)

Mapping Relationships (Parametrics)

Mapping Relationships (Parametrics)

mCAD model in SysML (assembly structure, properties, constraints)

eCAD model in SysML (key system-level entities and properties)

FEA models in SysML (analysis conditions & results)

System model in SysML

External tools and models

mCAD models (NX, Pro/E, CATIA,…)

eCAD models (Board Station, CR5000,…)

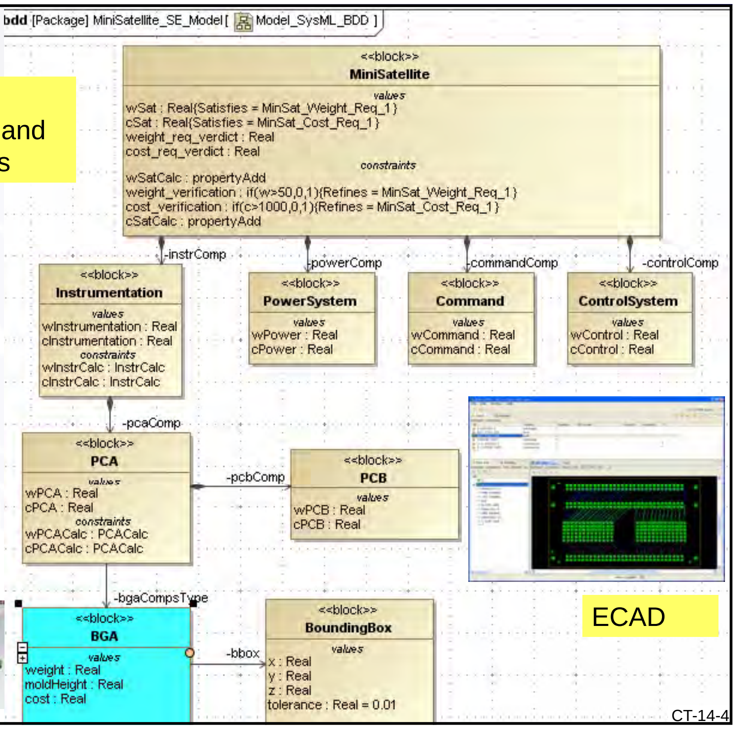
CAE models (FEA, CFD,…)

Other simulation models (STK, DEVS, …)

CT-14-3

CT-14.2

Connecting system model and domain models



bdd [Package] MiniSatellite_SE_Model [Model_SysML_BDD]

<<block>>
MiniSatellite
values
wSat : Real{Satisfies = MinSat_Weight_Req_1}
cSat : Real{Satisfies = MinSat_Cost_Req_1}
weight_req_verdict : Real
cost_req_verdict : Real
constraints
wSatCalc : propertyAdd
weight_verification : if(w>50,0,1){Refines = MinSat_Weight_Req_1}
cost_verification : if(c>1000,0,1){Refines = MinSat_Cost_Req_1}
cSatCalc : propertyAdd

-instrComp

-powerComp

-commandComp

-controlComp

<<block>>
Instrumentation
values
wInstrumentation : Real
cInstrumentation : Real
constraints
wInstrCalc : InstrCalc
cInstrCalc : InstrCalc

<<block>>
PowerSystem
values
wPower : Real
cPower : Real

<<block>>
Command
values
wCommand : Real
cCommand : Real

<<block>>
ControlSystem
values
wControl : Real
cControl : Real

-pcaComp

<<block>>
PCA
values
wPCA : Real
cPCA : Real
constraints
wPCACalc : PCACalc
cPCACalc : PCACalc

-pcbComp

<<block>>
PCB
values
wPCB : Real
cPCB : Real

-bgaCompsType

<<block>>
BGA
values
weight : Real
moldHeight : Real
cost : Real

-bbox

<<block>>
BoundingBox
values
x : Real
y : Real
z : Real
tolerance : Real = 0.01

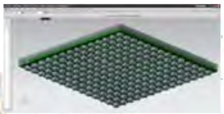
ECAD

PCA = printed circuit assembly

PCB = printed circuit board
(bare substrate w/ metal traces ...)

BGA = ball grid array
(a type of electronic component)

MCAD

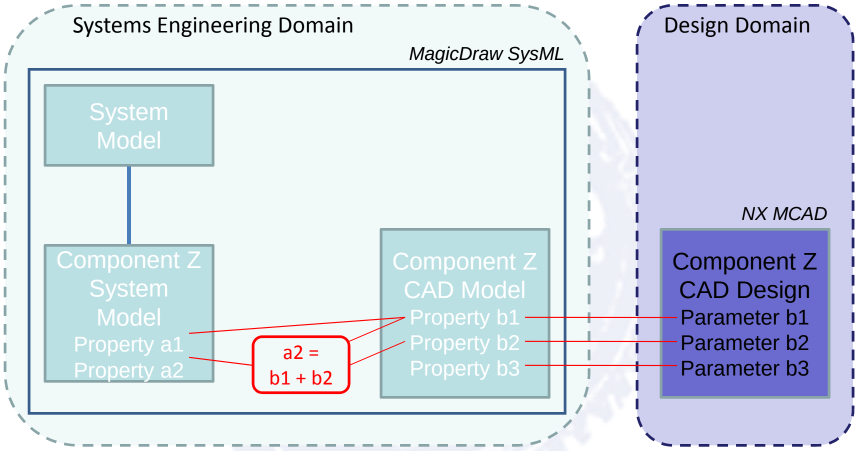


CT-14-4

CT-14.2

“System Model”- “X Domain Model” Integration

Ex. for X = Mechanical CAD

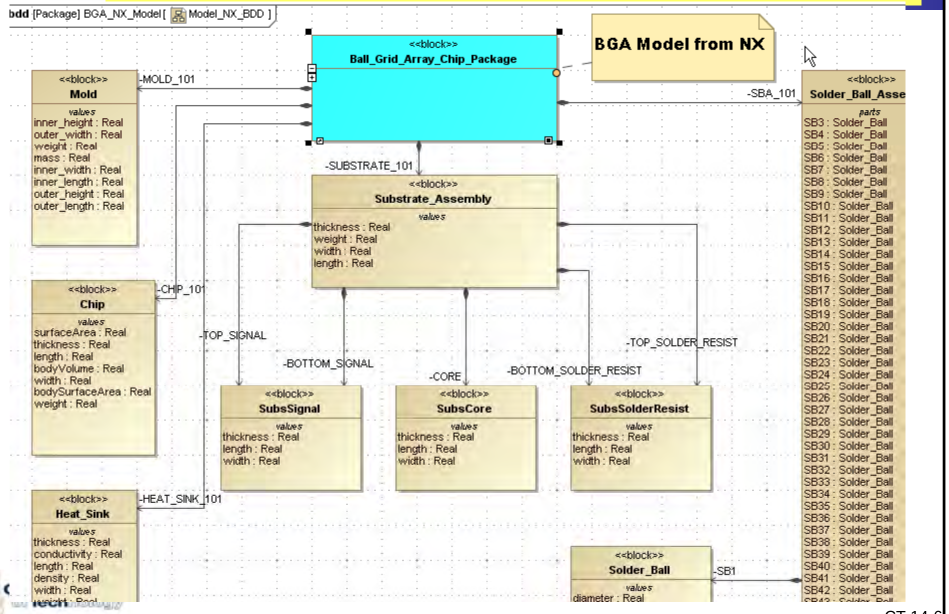


Step 1a Create a system model (e.g. with MagicDraw SysML)
Step 1b Create a CAD domain model (e.g. with Siemens NX)
Step 2 Import the CAD model into SysML as a CAD Model block
Step 3 Connect (map) the CAD model to the system model using SysML parametrics
Step 4 Control an auto-synch process: updates in CAD model ↔ updates in system model

CT-14-5

CT-14.2

SysML model of the BGA assembly automatically generated from NX MCAD model

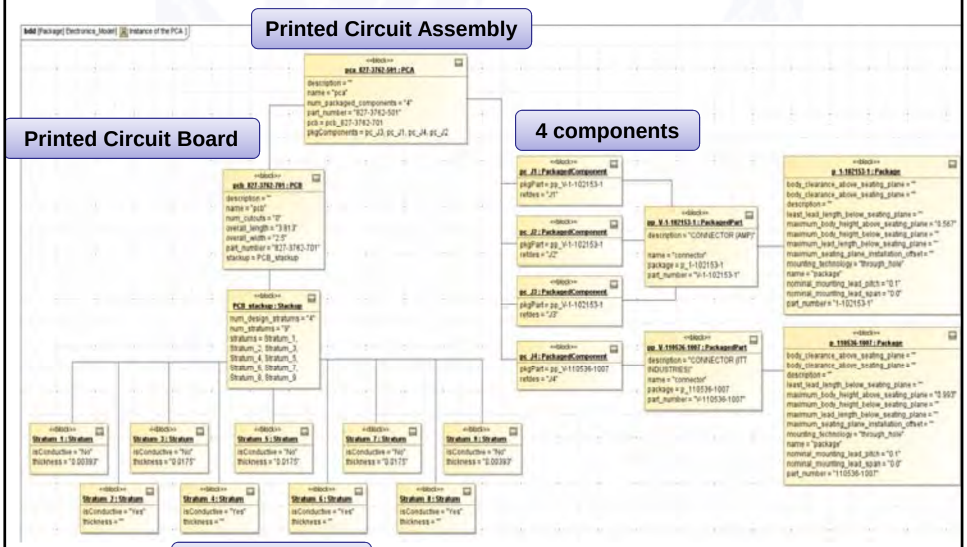


CT-14-6

CT-14.2

SysML Instance Model

Auto-generated from I-501 AP210 ECAD Model


bdd [Package] Electronics_Model [Instance of the PCA]
«block»
pca_827-3762-501 : PCA
description = ""
name = "pca"
num_packaged_components = "4"
part_number = "827-3762-501"
pcb = pcb_827-3762-701
pkgComponents = pc_J3, pc_J1, pc_J4, pc_J2
«block»
pcb_827-3762-701 : PCB
description = ""
name = "pcb"
num_cutouts = "0"
overall_length = "3.813"
overall_width = "2.5"
part_number = "827-3762-701"
stackup = PCB_stackup
«block»
PCB_stackup : Stackup
num_design_stratums = "4"
num_stratums = "9"
stratums = Stratum_1, Stratum_2, Stratum_3, Stratum_4, Stratum_5, Stratum_6, Stratum_7, Stratum_8, Stratum_9
«block»
pc_J1 : PackagedComponent
pkgPart = pp_V-1-102153-1
refdes = "J1"
«block»
pc_J2 : PackagedComponent
pkgPart = pp_V-1-102153-1
refdes = "J2"
«block»
pc_J3 : PackagedComponent
pkgPart = pp_V-1-102153-1
refdes = "J3"
«block»
pc_J4 : PackagedComponent
pkgPart = pp_V-110536-1007
refdes = "J4"
«block»
pp_V-1-102153-1 : PackagedPart
description = "CONNECTOR (AMP)"
name = "connector"
package = p_1-102153-1
part_number = "V-1-102153-1"
«block»
pp_V-110536-1007 : PackagedPart
description = "CONNECTOR (ITT INDUSTRIES)"
name = "connector"
package = p_110536-1007
part_number = "V-110536-1007"
«block»
p_1-102153-1 : Package
body_clearance_above_seating_plane = ""
body_clearance_above_seating_plane = ""
description = ""
least_lead_length_below_seating_plane = ""
maximum_body_height_above_seating_plane = "0.567"
maximum_body_height_below_seating_plane = ""
maximum_lead_length_below_seating_plane = ""
maximum_seating_plane_installation_offset = ""
mounting_technology = "through_hole"
name = "package"
nominal_mounting_lead_pitch = "0.1"
nominal_mounting_lead_span = "0.0"
part_number = "1-102153-1"
«block»
p_110536-1007 : Package
body_clearance_above_seating_plane = ""
body_clearance_above_seating_plane = ""
description = ""
least_lead_length_below_seating_plane = ""
maximum_body_height_above_seating_plane = "0.993"
maximum_body_height_below_seating_plane = ""
maximum_lead_length_below_seating_plane = ""
maximum_seating_plane_installation_offset = ""
mounting_technology = "through_hole"
name = "package"
nominal_mounting_lead_pitch = "0.1"
nominal_mounting_lead_span = "0.0"
part_number = "110536-1007"
«block»
Stratum_1 : Stratum
isConductive = "No"
thickness = "0.00393"
«block»
Stratum_3 : Stratum
isConductive = "No"
thickness = "0.0175"
«block»
Stratum_5 : Stratum
isConductive = "No"
thickness = "0.0175"
«block»
Stratum_7 : Stratum
isConductive = "No"
thickness = "0.0175"
«block»
Stratum_9 : Stratum
isConductive = "No"
thickness = "0.00393"
«block»
Stratum_2 : Stratum
isConductive = "Yes"
thickness = ""
«block»
Stratum_4 : Stratum
isConductive = "Yes"
thickness = ""
«block»
Stratum_6 : Stratum
isConductive = "Yes"
thickness = ""
«block»
Stratum_8 : Stratum
isConductive = "Yes"
thickness = ""

Printed Circuit Assembly

Printed Circuit Board

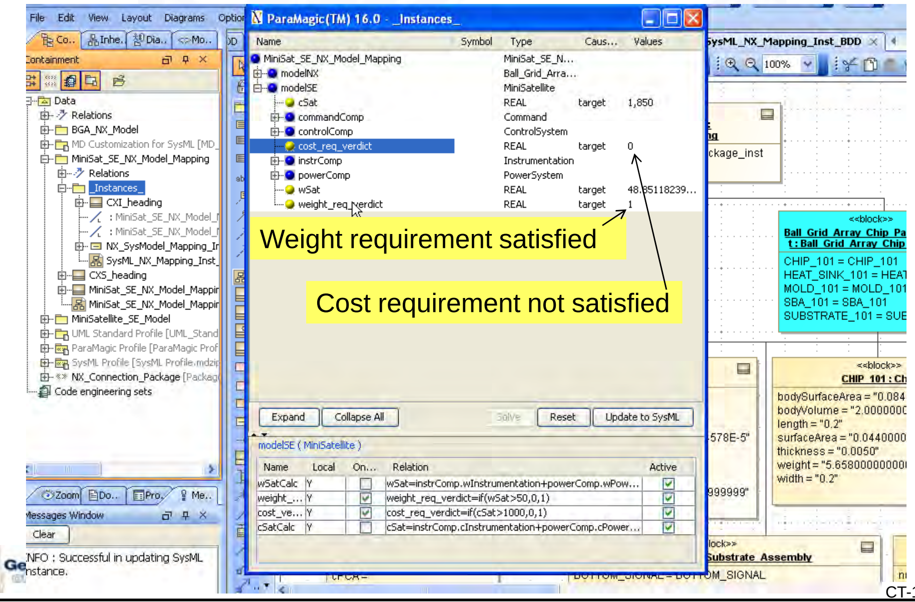
4 components

9 PCB stratums

Georgia Institute of Technology

CT-14-7

CT-14.2 ParaMagic is used to execute the resulting total model. It computes system-level cost & weight from all nested subsystem-level & component-level models (originating from M/ECAD/... tools), and it verifies related requirements.



Name	Symbol	Type	Caus...	Values
MiniSat_SE_NX_Model_Mapping		MiniSat_SE_N...		
modelNX		Ball_Grid_Arra...		
modelSE		MiniSatellite		
cSat		REAL	target	1,850
commandComp		Command		
controlComp		ControlSystem		
cost_req_verdict		REAL	target	0
instrComp		Instrumentation		
powerComp		PowerSystem		
wSat		REAL	target	48.85118239...
weight_req_verdict		REAL	target	1

Weight requirement satisfied

Cost requirement not satisfied

modelSE (MiniSatellite)

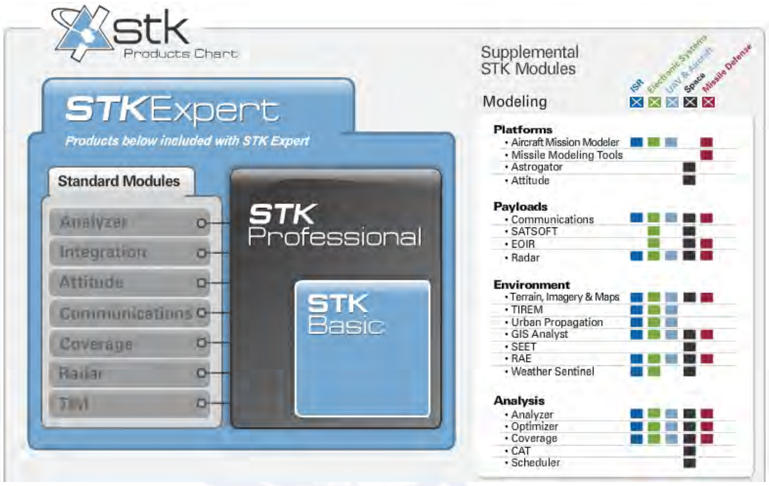
Name	Local	On...	Relation	Active
wSatCalc	Y	☐	wSat=instrComp.wInstrumentation+powerComp.wPow...	☑
weight_...	Y	☑	weight_req_verdict=if(wSat>50,0,1)	☑
cost_ve...	Y	☑	cost_req_verdict=if(cSat>1000,0,1)	☑
cSatCalc	Y	☐	cSat=instrComp.cInstrumentation+powerComp.cPower...	☑

INFO : Successful in updating SysML instance.

CT-14-8

CT-14.3: Live/Virtual/Constructive Simulations (LVC)

Representative Toolset: STK (www.agi.com)



Modeling	ISR	Electronic Systems	UAV & Aircraft	Space	Missile Defense
Platforms					
• Aircraft Mission Modeler	■	■	■		■
• Missile Modeling Tools					■
• Astrogator				■	
• Attitude				■	
Payloads					
• Communications	■	■	■	■	■
• SATSOFT		■		■	
• EOIR		■		■	■
• Radar	■	■	■	■	■
Environment					
• Terrain, Imagery & Maps	■	■	■	■	■
• TIREM	■	■	■		
• Urban Propagation	■	■	■		
• GIS Analyst	■	■	■	■	■
• SEET				■	
• RAE	■	■	■	■	■
• Weather Sentinel	■	■		■	
Analysis					
• Analyzer	■	■	■	■	■
• Optimizer	■	■	■	■	■
• Coverage	■	■	■	■	■
• CAT				■	
• Scheduler				■	

CT-14-9

CT-14.3: System/SoS M&S Examples in STK

Force-on-Force Fighter Simulation

(a) Normal model view

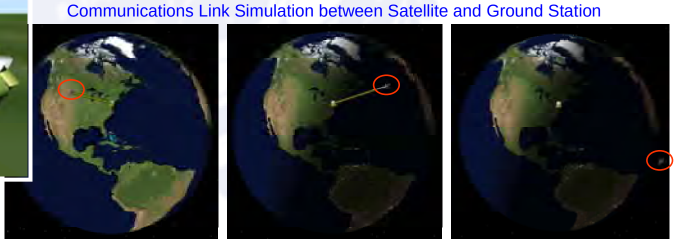
(b) Marker & trajectory history view

Geo-positioning Model

Missile Launcher Model

Mobile SA-10 Launcher
Latitude: 56.0
Longitude: 37.0

Communications Link Simulation between Satellite and Ground Station



(a) Link with ground station at t=t1

(b) Link with ground station at t=t2 (several orbits after t1)

(c) Link broken with ground station at t=t3 (~10 minutes after t2)

Based on original models by AGI.

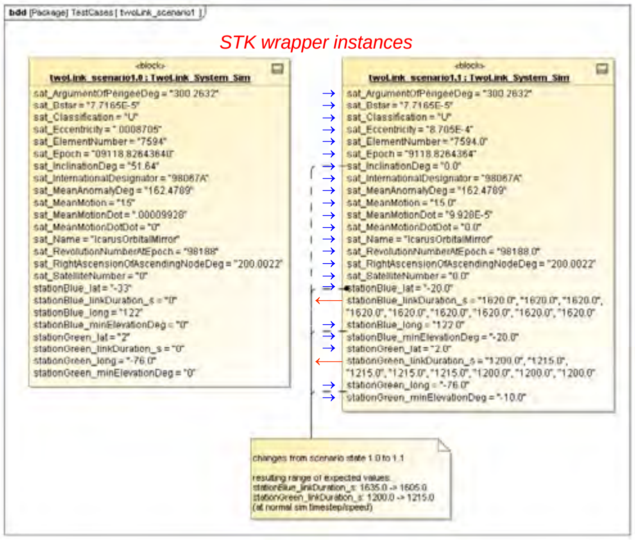
Georgia Institute of Technology

CT-14-10

CT-14.3: Two-way interoperability SysML-STK (throughout simulation run-time)

→ Changeable inputs (SysML to STK): satellite and ground station properties

← Results (STK to SysML): duration of ea. link session with ea. ground station



bdd [Package] TestCases [twoLink_scenario1]

STK wrapper instances

«block»
twoLink_scenario1.0 : TwoLink_System_Sim

sat_ArgumentOfPerigeeDeg = "300.2632"
sat_Bstar = "7.7165E-5"
sat_Classification = "U"
sat_Eccentricity = ".0008705"
sat_ElementNumber = "7594"
sat_Epoch = "09118.82643640"
sat_InclinationDeg = "51.64"
sat_InternationalDesignator = "98067A"
sat_MeanAnomalyDeg = "162.4789"
sat_MeanMotion = "15"
sat_MeanMotionDot = ".00009928"
sat_MeanMotionDotDot = "0"
sat_Name = "IcarusOrbitalMirror"
sat_RevolutionNumberAtEpoch = "98188"
sat_RightAscensionOfAscendingNodeDeg = "200.0022"
sat_SatelliteNumber = "0"
stationBlue_lat = "-33"
stationBlue_linkDuration_s = "0"
stationBlue_long = "122"
stationBlue_minElevationDeg = "0"
stationGreen_lat = "2"
stationGreen_linkDuration_s = "0"
stationGreen_long = "-76.0"
stationGreen_minElevationDeg = "0"

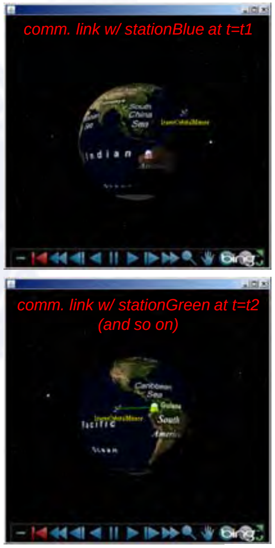
«block»
twoLink_scenario1.1 : TwoLink_System_Sim

→ sat_ArgumentOfPerigeeDeg = "300.2632"
→ sat_Bstar = "7.7165E-5"
→ sat_Classification = "U"
→ sat_Eccentricity = "8.705E-4"
→ sat_ElementNumber = "7594.0"
→ sat_Epoch = "9118.8264364"
→ sat_InclinationDeg = "0.0"
→ sat_InternationalDesignator = "98067A"
→ sat_MeanAnomalyDeg = "162.4789"
→ sat_MeanMotion = "15.0"
→ sat_MeanMotionDot = "9.928E-5"
→ sat_MeanMotionDotDot = "0.0"
→ sat_Name = "IcarusOrbitalMirror"
→ sat_RevolutionNumberAtEpoch = "98188.0"
→ sat_RightAscensionOfAscendingNodeDeg = "200.0022"
→ sat_SatelliteNumber = "0.0"
→ stationBlue_lat = "-20.0"
← stationBlue_linkDuration_s = "1620.0", "1620.0", "1620.0", "1620.0", "1620.0", "1620.0", "1620.0", "1620.0", "1620.0"
→ stationBlue_long = "122.0"
→ stationBlue_minElevationDeg = "-20.0"
→ stationGreen_lat = "2.0"
← stationGreen_linkDuration_s = "1200.0", "1215.0", "1215.0", "1215.0", "1215.0", "1200.0", "1200.0", "1200.0"
→ stationGreen_long = "-76.0"
→ stationGreen_minElevationDeg = "-10.0"

changes from scenario state 1.0 to 1.1

resulting range of expected values:
stationBlue_linkDuration_s: 1635.0 -> 1605.0
stationGreen_linkDuration_s: 1200.0 -> 1215.0
(at normal sim timestep/speed)

STK satellite comm. link sim
(a constructive simulation)

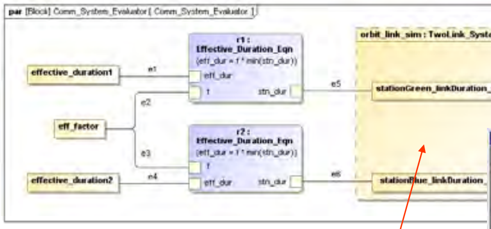


comm. link w/ stationBlue at t=t1

comm. link w/ stationGreen at t=t2
(and so on)

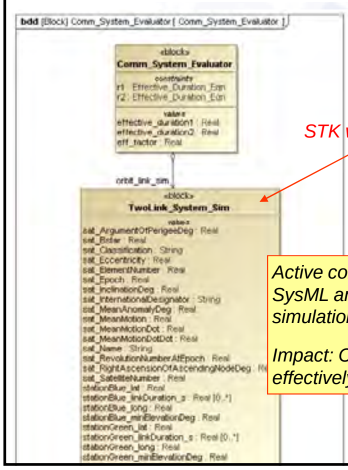
CT-14.3: Initial prototype STK & SysML parametrics

(for req. verification, ...)



par [Block] Comm_System_Evaluator [Comm_System_Evaluator]

effective_duration1
eff_factor
effective_duration2
e1
e2
e3
e4
e5
e6
r1 : Effective_Duration_Eqn
{eff_dur = f * min(stn_dur)}
eff_dur
f
stn_dur
r2 : Effective_Duration_Eqn
{eff_dur = f * min(stn_dur)}
f
eff_dur
stn_dur
orbit_link_sim : TwoLink_System_Sim
stationGreen_linkDuration_s [0..*]
stationBlue_linkDuration



bdd [Block] Comm_System_Evaluator [Comm_System_Evaluator]

«block»
Comm_System_Evaluator

constraints
r1 : Effective_Duration_Eqn
r2 : Effective_Duration_Eqn

values
effective_duration1 : Real
effective_duration2 : Real
eff_factor : Real

orbit_link_sim

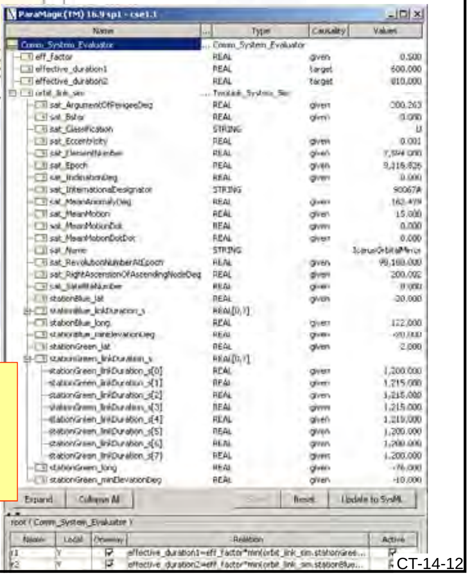
«block»
TwoLink_System_Sim

values
sat_ArgumentOfPerigeeDeg : Real
sat_Bstar : Real
sat_Classification : String
sat_Eccentricity : Real
sat_ElementNumber : Real
sat_Epoch : Real
sat_InclinationDeg : Real
sat_InternationalDesignator : String
sat_MeanAnomalyDeg : Real
sat_MeanMotion : Real
sat_MeanMotionDot : Real
sat_MeanMotionDotDot : Real
sat_Name : String
sat_RevolutionNumberAtEpoch : Real
sat_RightAscensionOfAscendingNodeDeg : Re
sat_SatelliteNumber : Real
stationBlue_lat : Real
stationBlue_linkDuration_s : Real [0..*]
stationBlue_long : Real
stationBlue_minElevationDeg : Real
stationGreen_lat : Real
stationGreen_linkDuration_s : Real [0..*]
stationGreen_long : Real
stationGreen_minElevationDeg : Real

STK wrapper block

Active connection between SysML and LVC-type simulations.

Impact: Can use SysML to effectively V&V such sims.



ParaMagic(TM) 16.9 sp1 - cse1.1

Name	...	Type	Causality	Values
Comm_System_Evaluator	...	Comm_System_Evaluator		
eff_factor		REAL	given	0.500
effective_duration1		REAL	target	600.000
effective_duration2		REAL	target	810.000
orbit_link_sim	...	TwoLink_System_Sim		
sat_ArgumentOfPerigeeDeg		REAL	given	300.263
sat_Bstar		REAL	given	0.000
sat_Classification		STRING		U
sat_Eccentricity		REAL	given	0.001
sat_ElementNumber		REAL	given	7,594.000
sat_Epoch		REAL	given	9,118.826
sat_InclinationDeg		REAL	given	0.000
sat_InternationalDesignator		STRING		98067A
sat_MeanAnomalyDeg		REAL	given	162.479
sat_MeanMotion		REAL	given	15.000
sat_MeanMotionDot		REAL	given	0.000
sat_MeanMotionDotDot		REAL	given	0.000
sat_Name		STRING		IcarusOrbitalMirror
sat_RevolutionNumberAtEpoch		REAL	given	98,188.000
sat_RightAscensionOfAscendingNodeDeg		REAL	given	200.002
sat_SatelliteNumber		REAL	given	0.000
stationBlue_lat		REAL	given	-20.000
stationBlue_linkDuration_s		REAL[0,*]		
stationBlue_long		REAL	given	122.000
stationBlue_minElevationDeg		REAL	given	-20.000
stationGreen_lat		REAL	given	2.000
stationGreen_linkDuration_s		REAL[0,*]		
stationGreen_linkDuration_s[0]		REAL	given	1,200.000
stationGreen_linkDuration_s[1]		REAL	given	1,215.000
stationGreen_linkDuration_s[2]		REAL	given	1,215.000
stationGreen_linkDuration_s[3]		REAL	given	1,215.000
stationGreen_linkDuration_s[4]		REAL	given	1,215.000
stationGreen_linkDuration_s[5]		REAL	given	1,200.000
stationGreen_linkDuration_s[6]		REAL	given	1,200.000
stationGreen_linkDuration_s[7]		REAL	given	1,200.000
stationGreen_long		REAL	given	-76.000
stationGreen_minElevationDeg		REAL	given	-10.000

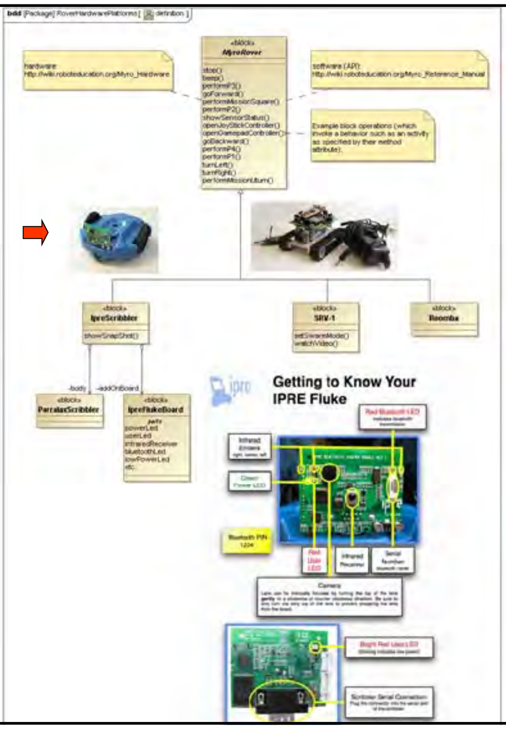
root (Comm_System_Evaluator)

Name	Local	Dummy	Relation	Active
r1	Y	☑	effective_duration1=eff_factor*min(orbit_link_sim.stationGree...	☑
r2	Y	☑	effective_duration2=eff_factor*min(orbit_link_sim.stationBlue...	☑

CT-15: Mobile Robot Context

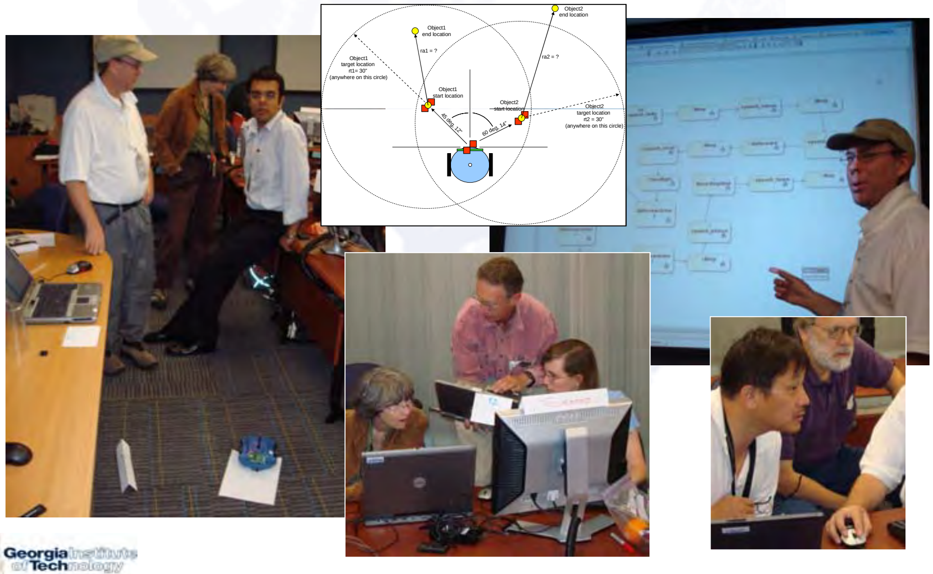
Myro rover
(a cyber-physical system)

CT-15: Towards automating verification and T&E of physical systems ...



SysML Activities Exercise @ JPL

Team Contest Using MyroMagic Plugin & Scribbler Rovers

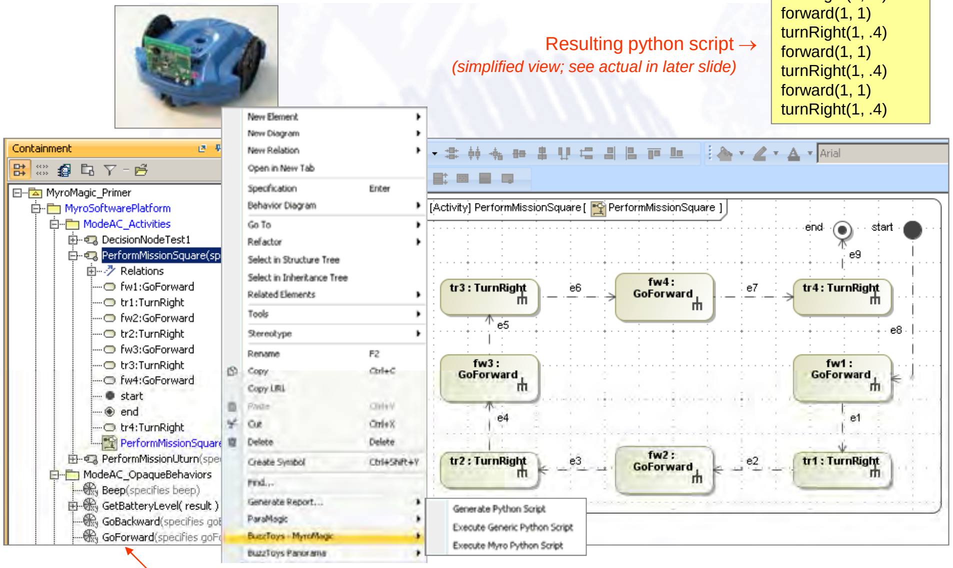


CT-15: Scribbler / MyroMagic Demo

Executable SysML Activity Model [1 - original]

```
from myro import *
initialize("com29")

forward(1, 1)
turnRight(1, .4)
forward(1, 1)
turnRight(1, .4)
forward(1, 1)
turnRight(1, .4)
forward(1, 1)
turnRight(1, .4)
```

Resulting python script →
(simplified view; see actual in later slide)



opaque behaviors (native code segments)

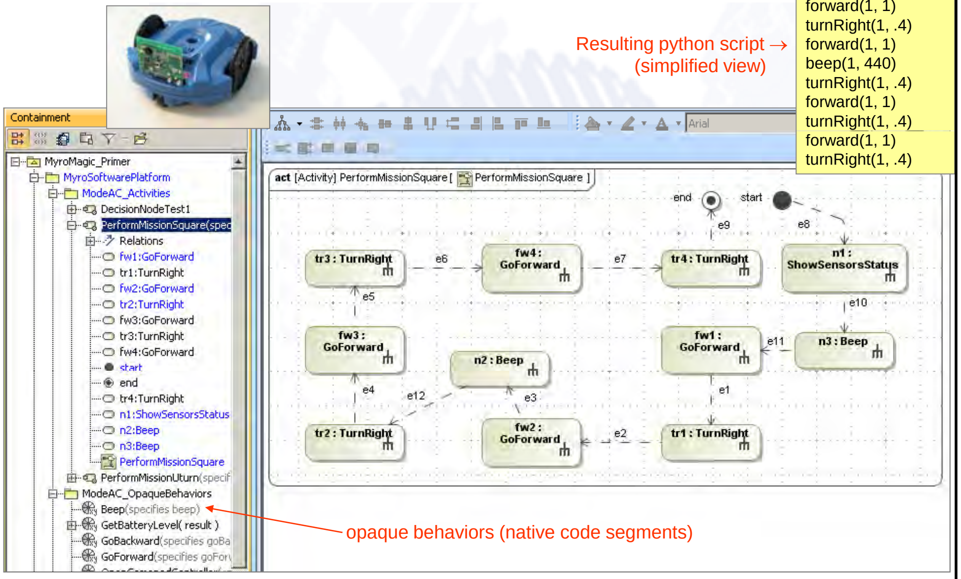
CT-15-3

CT-15: Scribbler / MyroMagic Demo

Executable SysML Activity Model [2 - after live update]

```
from myro import *
initialize("com29")

senses()
beep(1, 440)
forward(1, 1)
turnRight(1, .4)
forward(1, 1)
beep(1, 440)
turnRight(1, .4)
forward(1, 1)
turnRight(1, .4)
forward(1, 1)
turnRight(1, .4)
```

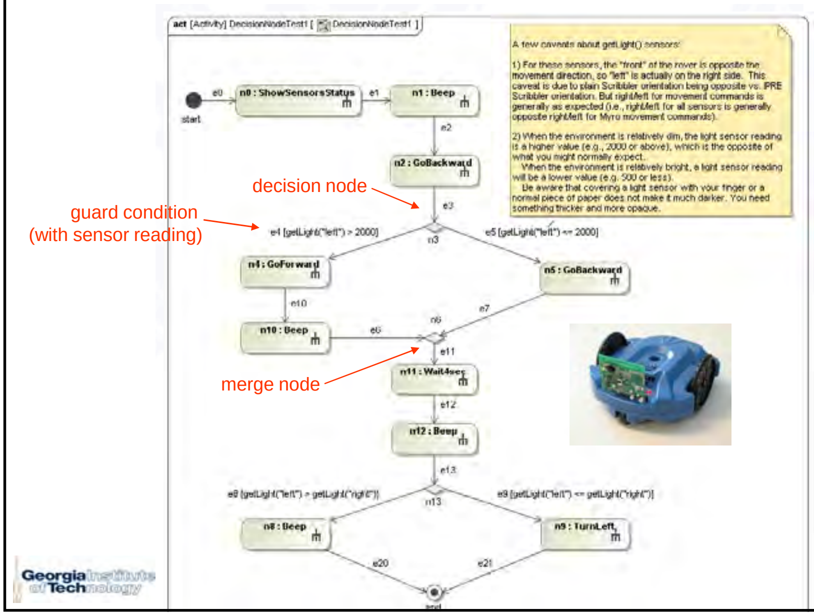
Resulting python script →
(simplified view)



opaque behaviors (native code segments)

CT-15-4

CT-15: (cont.)
Choosing activity path based on sensor readings



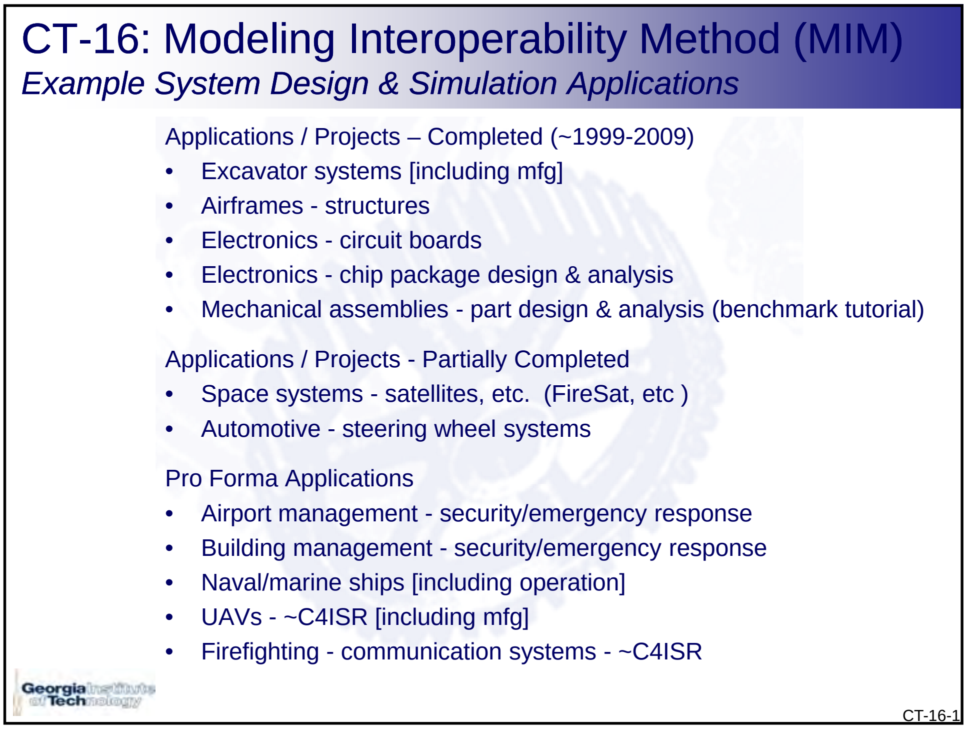
A few caveats about getLight() sensors:

1) For these sensors, the "front" of the rover is opposite the movement direction, so "left" is actually on the right side. This caveat is due to plain Scribbler orientation being opposite vs. IPRE Scribbler orientation. But right/left for movement commands is generally as expected (i.e., right/left for all sensors is generally opposite right/left for Myro movement commands).

2) When the environment is relatively dim, the light sensor reading is a higher value (e.g., 2000 or above), which is the opposite of what you might normally expect.
When the environment is relatively bright, a light sensor reading will be a lower value (e.g. 500 or less).
Be aware that covering a light sensor with your finger or a normal piece of paper does not make it much darker. You need something thicker and more opaque.

[Part 3] Contents

- Core embedded V&V concepts [CT-1 to CT-5]
- Higher-level concepts – round1 [CT-6 to CT-11]
  - Sample SysML-based V&V building blocks
  - Example applications
- Higher-level concepts – round2 [CT-12 to CT-15]
  - Verifying external models & systems via SysML
- ➡ Higher-level concepts – round3 [CT-16 to CT-17]
  - MIM architecture for M&S patterns
  - Other concept extensions



CT-16: Modeling Interoperability Method (MIM)

Example System Design & Simulation Applications

Applications / Projects – Completed (~1999-2009)

- Excavator systems [including mfg]
- Airframes - structures
- Electronics - circuit boards
- Electronics - chip package design & analysis
- Mechanical assemblies - part design & analysis (benchmark tutorial)

Applications / Projects - Partially Completed

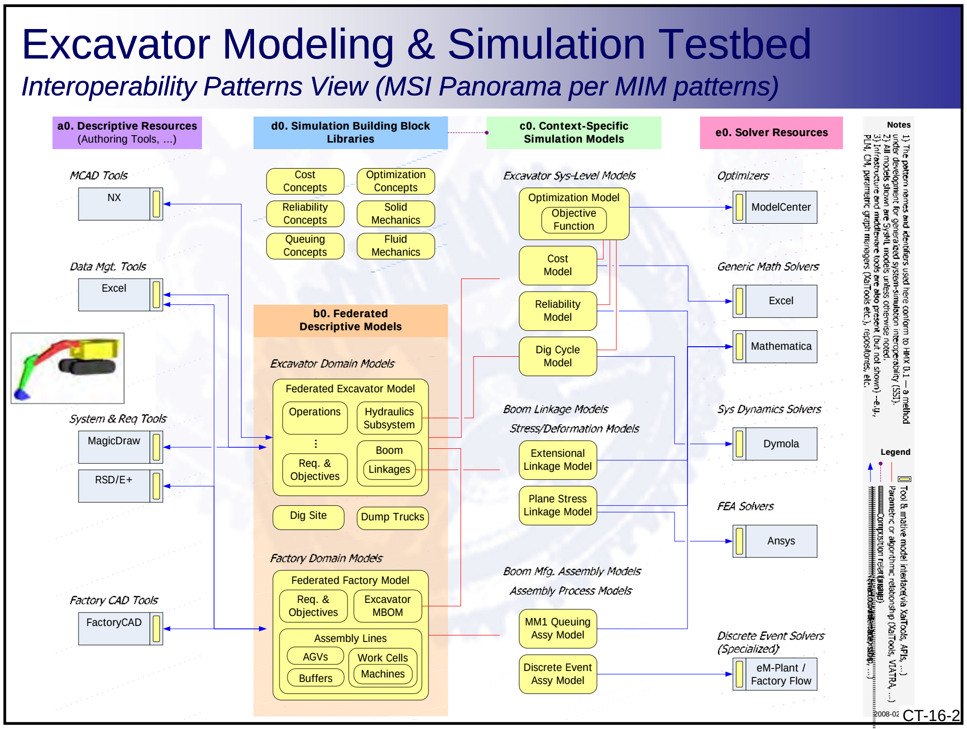
- Space systems - satellites, etc. (FireSat, etc)
- Automotive - steering wheel systems

Pro Forma Applications

- Airport management - security/emergency response
- Building management - security/emergency response
- Naval/marine ships [including operation]
- UAVs - ~C4ISR [including mfg]
- Firefighting - communication systems - ~C4ISR

Georgia Institute of Technology

CT-16-1



Excavator Modeling & Simulation Testbed

Interoperability Patterns View (MSI Panorama per MIM patterns)

a0. Descriptive Resources (Authoring Tools, ...)

MCAD Tools
NX

Data Mgt. Tools
Excel

System & Req Tools
MagicDraw
RSD/E+

Factory CAD Tools
FactoryCAD

d0. Simulation Building Block Libraries

Cost Concepts
Optimization Concepts
Reliability Concepts
Solid Mechanics
Queuing Concepts
Fluid Mechanics

b0. Federated Descriptive Models

Excavator Domain Models
Federated Excavator Model
Operations
Hydraulics Subsystem
⋮
Boom
Req. & Objectives
Linkages
Dig Site
Dump Trucks

Factory Domain Models
Federated Factory Model
Req. & Objectives
Excavator MBOM
Assembly Lines
AGVs
Work Cells
Buffers
Machines

c0. Context-Specific Simulation Models

Excavator Sys-Level Models
Optimization Model
Objective Function
Cost Model
Reliability Model
Dig Cycle Model

Boom Linkage Models
Stress/Deformation Models
Extensional Linkage Model
Plane Stress Linkage Model

Boom Mfg. Assembly Models
Assembly Process Models
MM1 Queuing Assy Model
Discrete Event Assy Model

e0. Solver Resources

Optimizers
ModelCenter

Generic Math Solvers
Excel
Mathematica

Sys Dynamics Solvers
Dymola

FEA Solvers
Ansys

Discrete Event Solvers (Specialized)
eM-Plant / Factory Flow

Notes

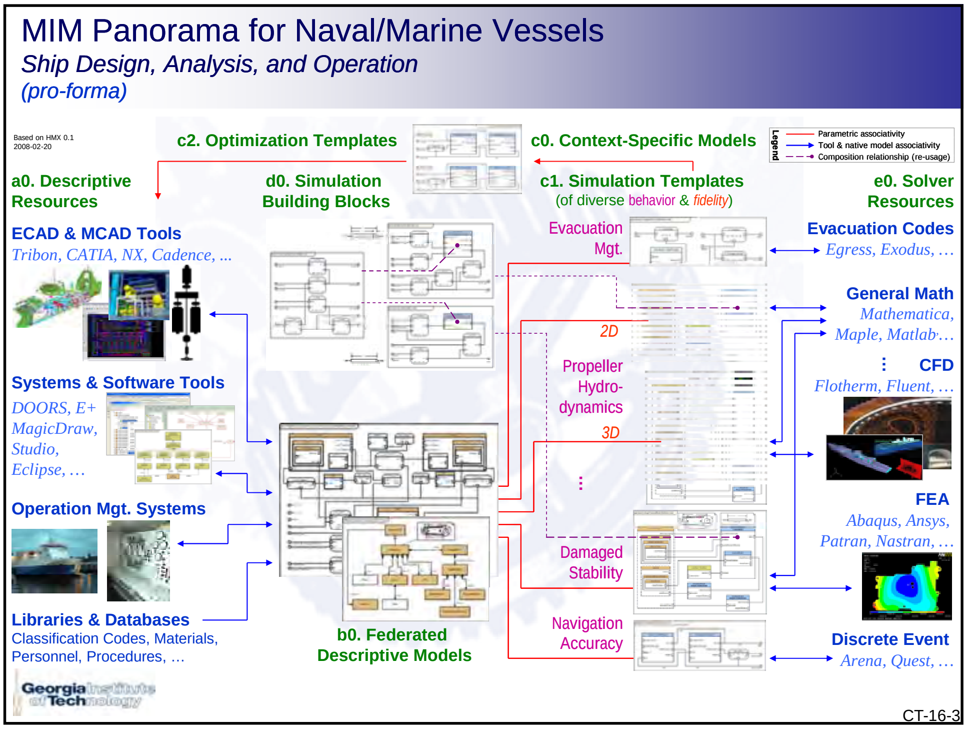
1) The pattern names and identifiers used here conform to HMX 0.1 — a method under development for generalized system-simulation interoperability (SSI).
2) All models shown are SysML models unless otherwise noted.
3) Infrastructure and middleware tools are also present (but not shown) --e.g., PLM, CM, parametric graph managers (XaiTools etc.), repositories, etc.

Legend

Tool & native model interface (via XaiTools, APIs, ...)
Parametric or algorithmic relationship (XaiTools, VIATRA, ...)
Composition relationship

2008-02

CT-16-2



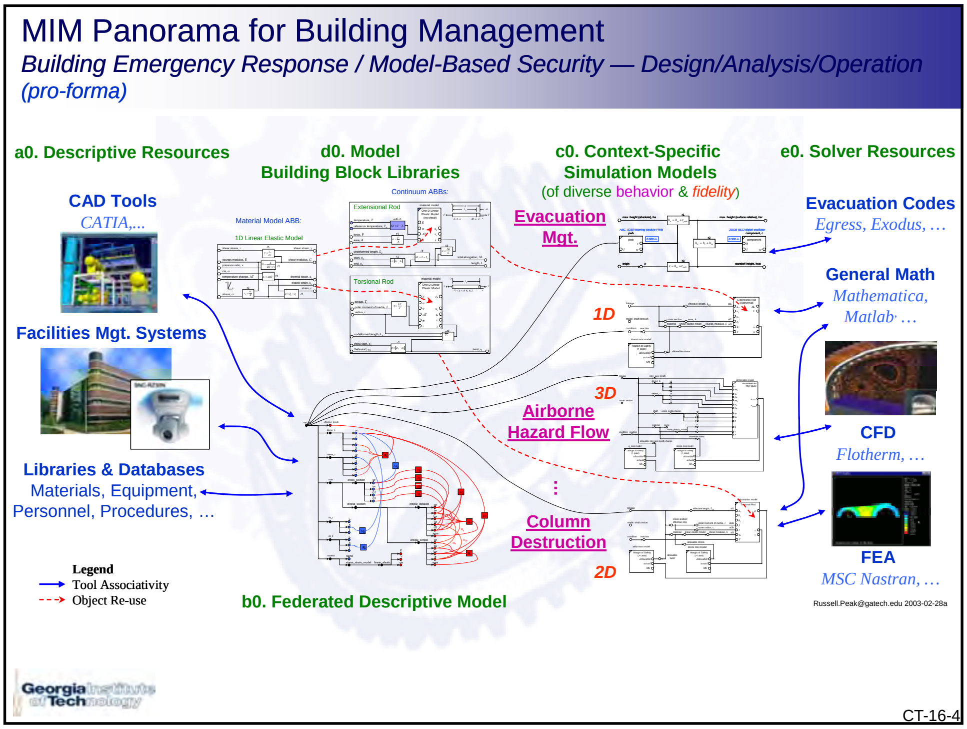
MIM Panorama for Naval/Marine Vessels

Ship Design, Analysis, and Operation
(pro-forma)

Based on HMX 0.1
2008-02-20

c2. Optimization Templates
c0. Context-Specific Models
Legend
Parametric associativity
Tool & native model associativity
Composition relationship (re-usage)
a0. Descriptive Resources
d0. Simulation Building Blocks
c1. Simulation Templates
(of diverse behavior & fidelity)
e0. Solver Resources
ECAD & MCAD Tools
Tribon, CATIA, NX, Cadence, ...
Evacuation Mgt.
Evacuation Codes
Egress, Exodus, …
General Math
Mathematica, Maple, Matlab,…
2D
⋮
CFD
Flotherm, Fluent, …
Systems & Software Tools
DOORS, E+ MagicDraw, Studio, Eclipse, …
Propeller Hydro-dynamics
3D
⋮
FEA
Abaqus, Ansys, Patran, Nastran, …
Operation Mgt. Systems
Damaged Stability
Libraries & Databases
Classification Codes, Materials, Personnel, Procedures, …
b0. Federated Descriptive Models
Navigation Accuracy
Discrete Event
Arena, Quest, …
Georgia Institute of Technology

CT-16-3



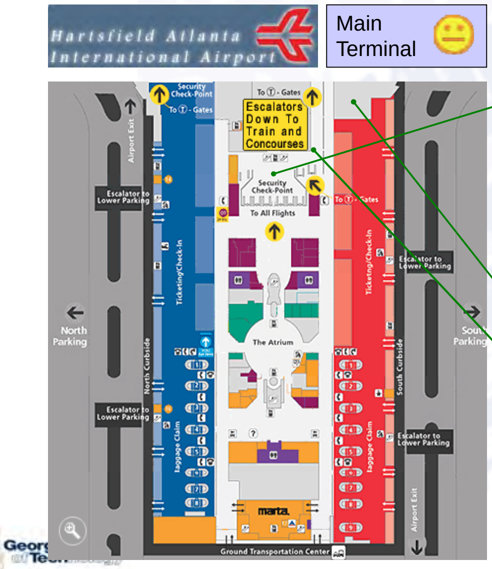
MIM Panorama for Building Management

Building Emergency Response / Model-Based Security — Design/Analysis/Operation
(pro-forma)

CT-16-4

Security Center Dashboard

Overall Status of Key Systems



Main Terminal 😐

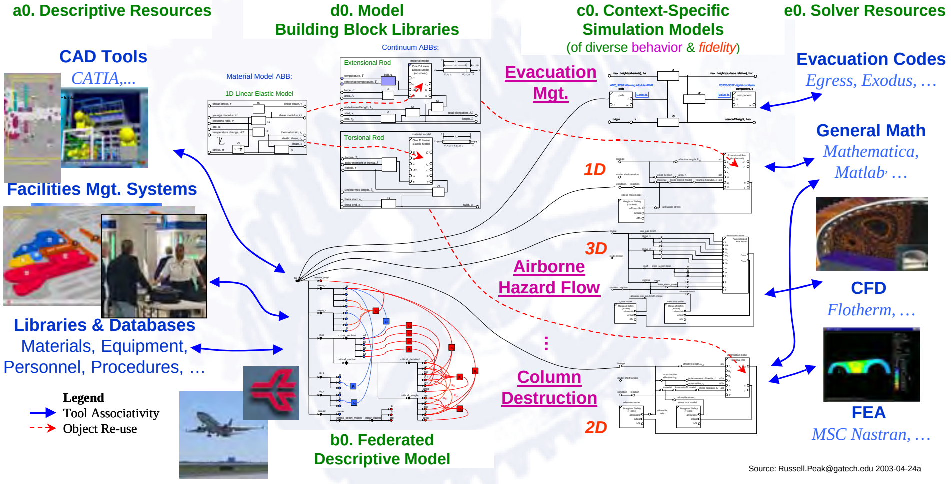
System	*Status*
Crowd Controls	😐
Passenger Screening	☺
Cargo Screening	☺
Communication Systems	☺
Electrical Systems	☺
HVAC Systems	😐
Chemical Detection	☹
Biological Detection	☺
…	…

Source: Russell.Peak@marc.gatech.edu 2003-04-24 CT-16-5

MIM Panorama for Airport Management

Airport Emergency Response / Model-Based Security — Design/Analysis/Operation

ATL / Hartsfield Jackson International Airport (HJIA) (pro-forma)



a0. Descriptive Resources

d0. Model Building Block Libraries

c0. Context-Specific Simulation Models (of diverse behavior & *fidelity*)

e0. Solver Resources

CAD Tools *CATIA,...*

Facilities Mgt. Systems

Libraries & Databases Materials, Equipment, Personnel, Procedures, …

Continuum ABBs:

Material Model ABB:

1D Linear Elastic Model

Extensional Rod

Torsional Rod

Evacuation Mgt.

1D

3D

Airborne Hazard Flow

:

Column Destruction

2D

Evacuation Codes *Egress, Exodus, …*

General Math *Mathematica, Matlab, …*

CFD *Flotherm, …*

FEA *MSC Nastran, …*

b0. Federated Descriptive Model

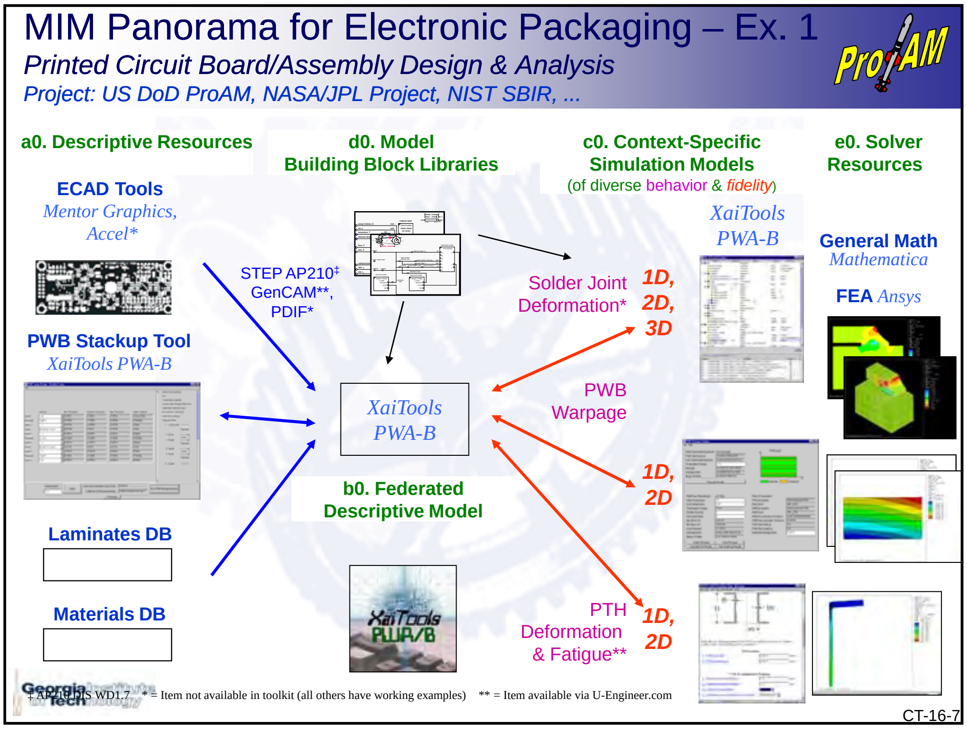
Legend

Tool Associativity

Object Re-use

Source: Russell.Peak@gatech.edu 2003-04-24a

CT-16-6



MIM Panorama for Electronic Packaging – Ex. 1

Printed Circuit Board/Assembly Design & Analysis

Project: US DoD ProAM, NASA/JPL Project, NIST SBIR, ...

a0. Descriptive Resources

d0. Model Building Block Libraries

c0. Context-Specific Simulation Models (of diverse behavior & *fidelity*)

e0. Solver Resources

ECAD Tools
*Mentor Graphics, Accel**

STEP AP210‡
GenCAM**,
PDIF*

PWB Stackup Tool
XaiTools PWA-B

Laminates DB

Materials DB

XaiTools PWA-B

b0. Federated Descriptive Model

Solder Joint Deformation* **1D, 2D, 3D**

PWB Warpage **1D, 2D**

PTH Deformation & Fatigue** **1D, 2D**

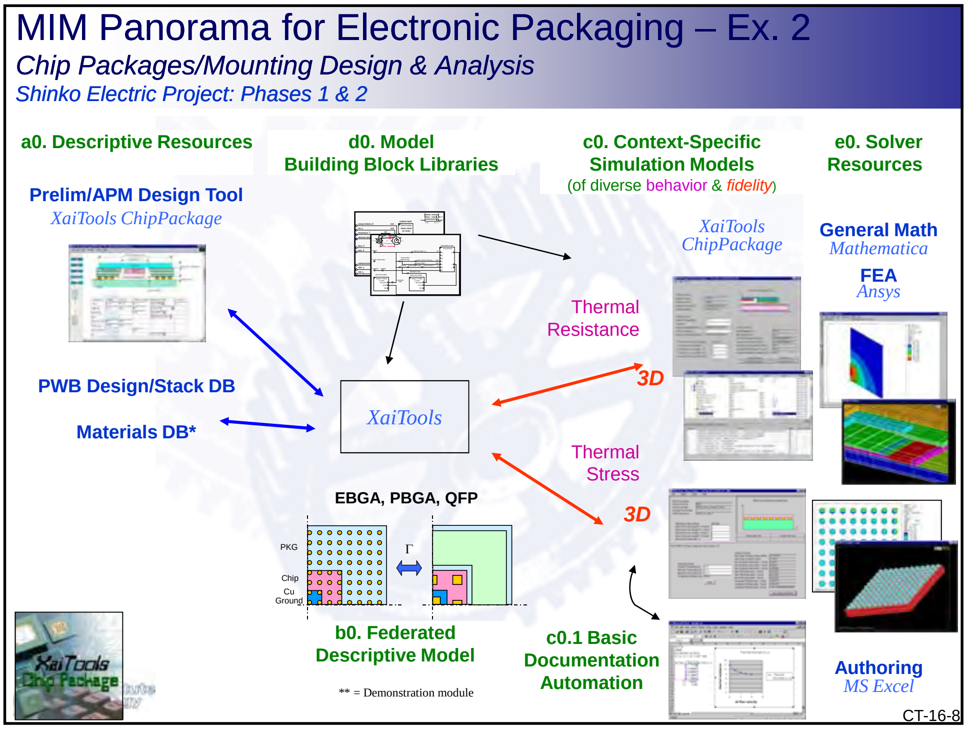
XaiTools PWA-B

General Math *Mathematica*

FEA *Ansys*

‡ AP210 DIS WD1.7 * = Item not available in toolkit (all others have working examples) ** = Item available via U-Engineer.com

CT-16-7



MIM Panorama for Electronic Packaging – Ex. 2

Chip Packages/Mounting Design & Analysis

Shinko Electric Project: Phases 1 & 2

a0. Descriptive Resources

d0. Model Building Block Libraries

c0. Context-Specific Simulation Models (of diverse behavior & *fidelity*)

e0. Solver Resources

Prelim/APM Design Tool
XaiTools ChipPackage

PWB Design/Stack DB

Materials DB*

XaiTools

EBGA, PBGA, QFP

b0. Federated Descriptive Model

** = Demonstration module

Thermal Resistance **3D**

Thermal Stress **3D**

c0.1 Basic Documentation Automation

XaiTools ChipPackage

General Math *Mathematica*

FEA *Ansys*

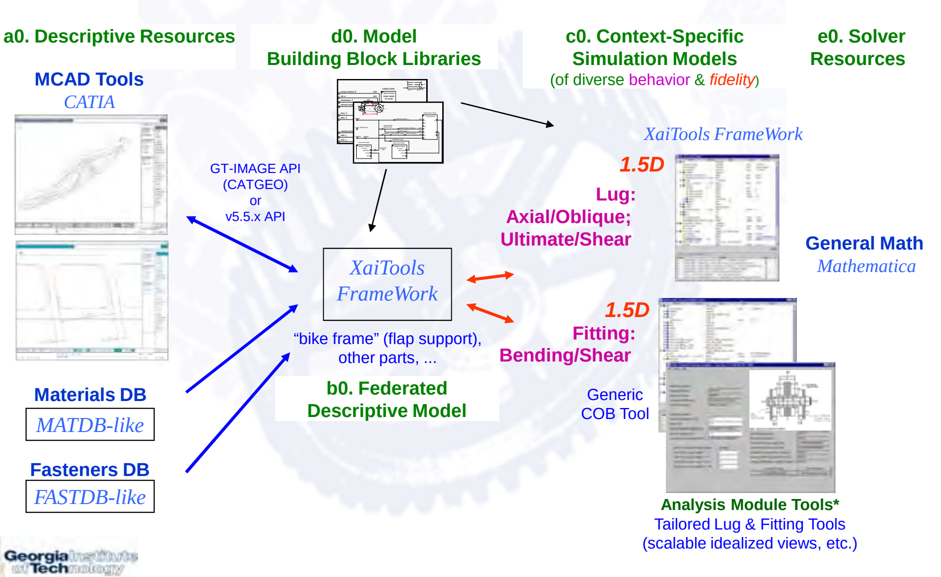
Authoring *MS Excel*

CT-16-8

MIM Panorama for Airframe Structures – Ex. 2

Lug & Fitting Design & Analysis

Boeing PSI Phases 3.0 and 3.1 (pro forma)



a0. Descriptive Resources

d0. Model Building Block Libraries

c0. Context-Specific Simulation Models (of diverse behavior & *fidelity*)

e0. Solver Resources

MCAD Tools
CATIA

GT-IMAGE API (CATGEO) or v5.5.x API

XaiTools FrameWork

"bike frame" (flap support), other parts, ...

b0. Federated Descriptive Model

Materials DB
MATDB-like

Fasteners DB
FASTDB-like

XaiTools FrameWork

1.5D
Lug: Axial/Oblique; Ultimate/Shear

1.5D
Fitting: Bending/Shear

Generic COB Tool

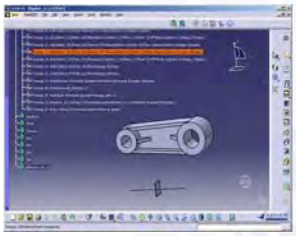
General Math
Mathematica

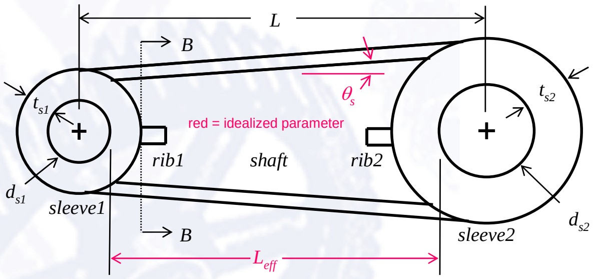
Analysis Module Tools*
Tailored Lug & Fitting Tools
(scalable idealized views, etc.)

CT-16-9

Flap Linkage Mechanical Part

A simple design ... a benchmark problem.





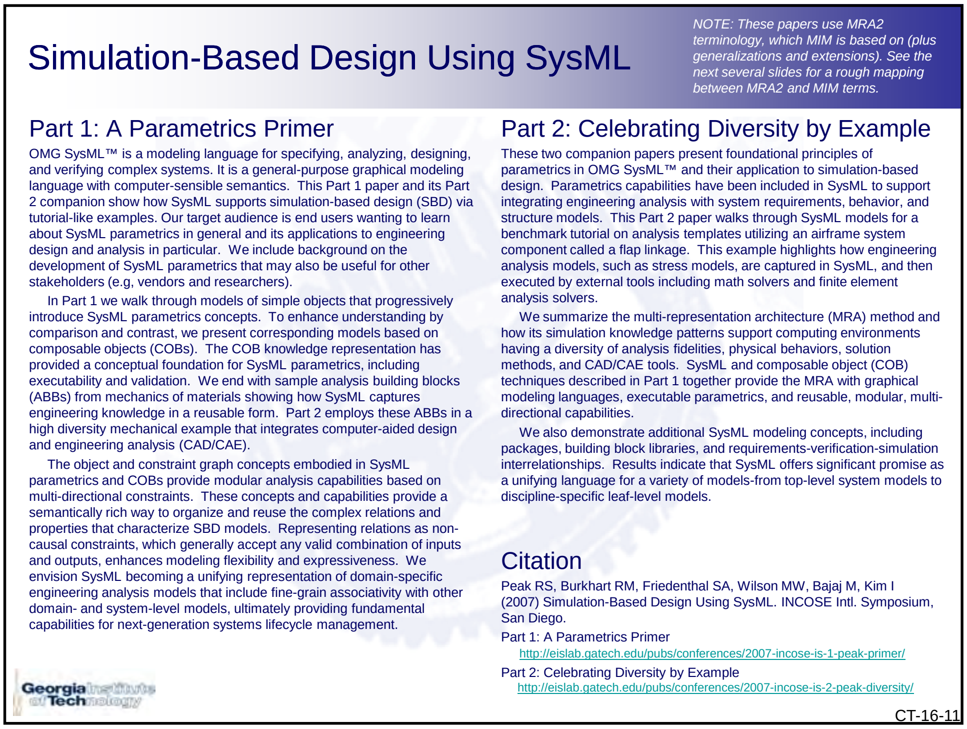
Background

This simple part provides the basis for a benchmark tutorial for CAD-CAE interoperability and simulation template knowledge representation. This example exercises multiple capabilities relevant to such contexts (many of which are relevant to broader simulation and knowledge representation domains), including:

- Diversity in design information source, behavior, fidelity, solution method, solution tool, ...
- Modular, reusable simulation building blocks and fine-grained inter-model associativity

See the following for further information:
- http://eislab.gatech.edu/pubs/conferences/2007-incose-is-1-peak-primer/
- http://eislab.gatech.edu/pubs/conferences/2007-incose-is-2-peak-diversity/

CT-16-10



Simulation-Based Design Using SysML

NOTE: These papers use MRA2 terminology, which MIM is based on (plus generalizations and extensions). See the next several slides for a rough mapping between MRA2 and MIM terms.

Part 1: A Parametrics Primer

OMG SysML™ is a modeling language for specifying, analyzing, designing, and verifying complex systems. It is a general-purpose graphical modeling language with computer-sensible semantics. This Part 1 paper and its Part 2 companion show how SysML supports simulation-based design (SBD) via tutorial-like examples. Our target audience is end users wanting to learn about SysML parametrics in general and its applications to engineering design and analysis in particular. We include background on the development of SysML parametrics that may also be useful for other stakeholders (e.g, vendors and researchers).

In Part 1 we walk through models of simple objects that progressively introduce SysML parametrics concepts. To enhance understanding by comparison and contrast, we present corresponding models based on composable objects (COBs). The COB knowledge representation has provided a conceptual foundation for SysML parametrics, including executability and validation. We end with sample analysis building blocks (ABBs) from mechanics of materials showing how SysML captures engineering knowledge in a reusable form. Part 2 employs these ABBs in a high diversity mechanical example that integrates computer-aided design and engineering analysis (CAD/CAE).

The object and constraint graph concepts embodied in SysML parametrics and COBs provide modular analysis capabilities based on multi-directional constraints. These concepts and capabilities provide a semantically rich way to organize and reuse the complex relations and properties that characterize SBD models. Representing relations as non-causal constraints, which generally accept any valid combination of inputs and outputs, enhances modeling flexibility and expressiveness. We envision SysML becoming a unifying representation of domain-specific engineering analysis models that include fine-grain associativity with other domain- and system-level models, ultimately providing fundamental capabilities for next-generation systems lifecycle management.

Part 2: Celebrating Diversity by Example

These two companion papers present foundational principles of parametrics in OMG SysML™ and their application to simulation-based design. Parametrics capabilities have been included in SysML to support integrating engineering analysis with system requirements, behavior, and structure models. This Part 2 paper walks through SysML models for a benchmark tutorial on analysis templates utilizing an airframe system component called a flap linkage. This example highlights how engineering analysis models, such as stress models, are captured in SysML, and then executed by external tools including math solvers and finite element analysis solvers.

We summarize the multi-representation architecture (MRA) method and how its simulation knowledge patterns support computing environments having a diversity of analysis fidelities, physical behaviors, solution methods, and CAD/CAE tools. SysML and composable object (COB) techniques described in Part 1 together provide the MRA with graphical modeling languages, executable parametrics, and reusable, modular, multi-directional capabilities.

We also demonstrate additional SysML modeling concepts, including packages, building block libraries, and requirements-verification-simulation interrelationships. Results indicate that SysML offers significant promise as a unifying language for a variety of models-from top-level system models to discipline-specific leaf-level models.

Citation

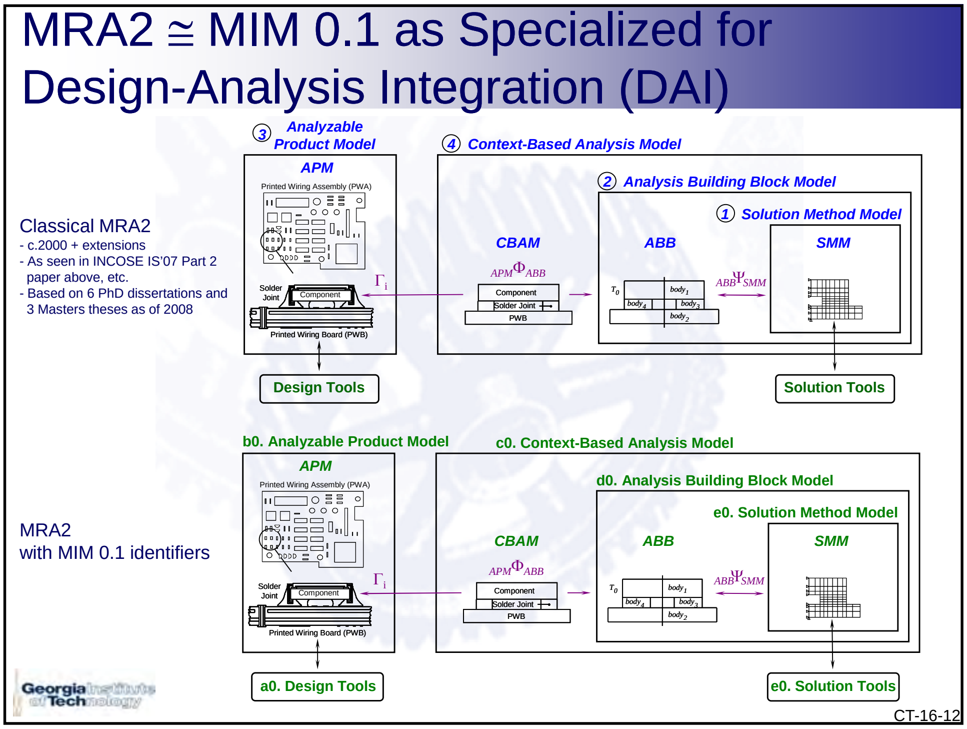
Peak RS, Burkhart RM, Friedenthal SA, Wilson MW, Bajaj M, Kim I (2007) Simulation-Based Design Using SysML. INCOSE Intl. Symposium, San Diego.

Part 1: A Parametrics Primer
http://eislab.gatech.edu/pubs/conferences/2007-incose-is-1-peak-primer/

Part 2: Celebrating Diversity by Example
http://eislab.gatech.edu/pubs/conferences/2007-incose-is-2-peak-diversity/

Georgia Institute of Technology

CT-16-11



MRA2 ≅ MIM 0.1 as Specialized for Design-Analysis Integration (DAI)

Classical MRA2
- c.2000 + extensions
- As seen in INCOSE IS'07 Part 2 paper above, etc.
- Based on 6 PhD dissertations and 3 Masters theses as of 2008

MRA2
with MIM 0.1 identifiers

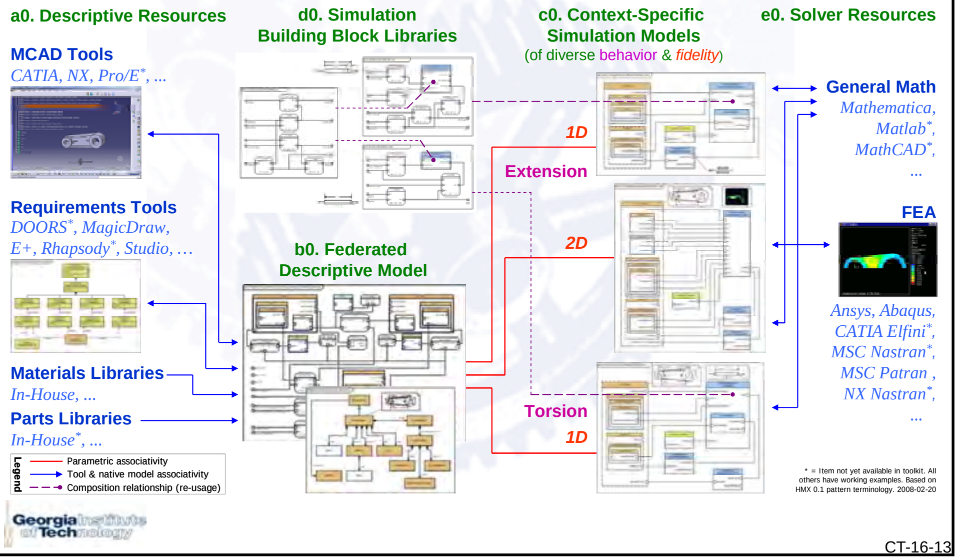
Georgia Institute of Technology

CT-16-12

MIM Panorama for Mechanical Design

Flap Linkage Model—A Benchmark Design-Analysis Example

Interoperability Panorama View (per MIM 0.1 terminology)



a0. Descriptive Resources

MCAD Tools
CATIA, NX, Pro/E, …*

Requirements Tools
DOORS, MagicDraw, E+, Rhapsody*, Studio, …*

Materials Libraries
In-House, …

Parts Libraries
In-House, …*

d0. Simulation Building Block Libraries

b0. Federated Descriptive Model

c0. Context-Specific Simulation Models (of diverse behavior & *fidelity*)

e0. Solver Resources

General Math
Mathematica, Matlab, MathCAD*, …*

FEA
Ansys, Abaqus, CATIA Elfini, MSC Nastran*, MSC Patran , NX Nastran*, …*

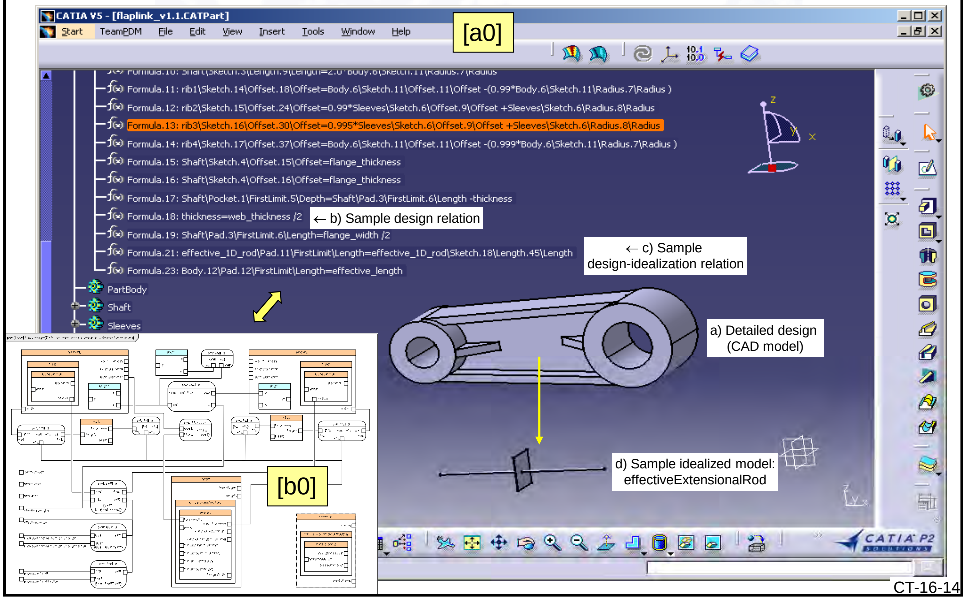
Legend
- Parametric associativity
- Tool & native model associativity
- Composition relationship (re-usage)

* = Item not yet available in toolkit. All others have working examples. Based on HMX 0.1 pattern terminology. 2008-02-20

Georgia Institute of Technology

CT-16-13

a0 FlapLinkage Implementation in CATIA v5



```
Formula.10: Shaft\Sketch.3\Length.9\Length=2.0*Body.6\Sketch.11\Radius.7\Radius
Formula.11: rib1\Sketch.14\Offset.18\Offset=Body.6\Sketch.11\Offset.11\Offset -(0.99*Body.6\Sketch.11\Radius.7\Radius )
Formula.12: rib2\Sketch.15\Offset.24\Offset=0.99*Sleeves\Sketch.6\Offset.9\Offset +Sleeves\Sketch.6\Radius.8\Radius
Formula.13: rib3\Sketch.16\Offset.30\Offset=0.995*Sleeves\Sketch.6\Offset.9\Offset +Sleeves\Sketch.6\Radius.8\Radius
Formula.14: rib4\Sketch.17\Offset.37\Offset=Body.6\Sketch.11\Offset.11\Offset -(0.999*Body.6\Sketch.11\Radius.7\Radius )
Formula.15: Shaft\Sketch.4\Offset.15\Offset=flange_thickness
Formula.16: Shaft\Sketch.4\Offset.16\Offset=flange_thickness
Formula.17: Shaft\Pocket.1\FirstLimit.5\Depth=Shaft\Pad.3\FirstLimit.6\Length -thickness
Formula.18: thickness=web_thickness /2
Formula.19: Shaft\Pad.3\FirstLimit.6\Length=flange_width /2
Formula.21: effective_1D_rod\Pad.11\FirstLimit\Length=effective_1D_rod\Sketch.18\Length.45\Length
Formula.23: Body.12\Pad.12\FirstLimit\Length=effective_length
```

← b) Sample design relation

← c) Sample design-idealization relation

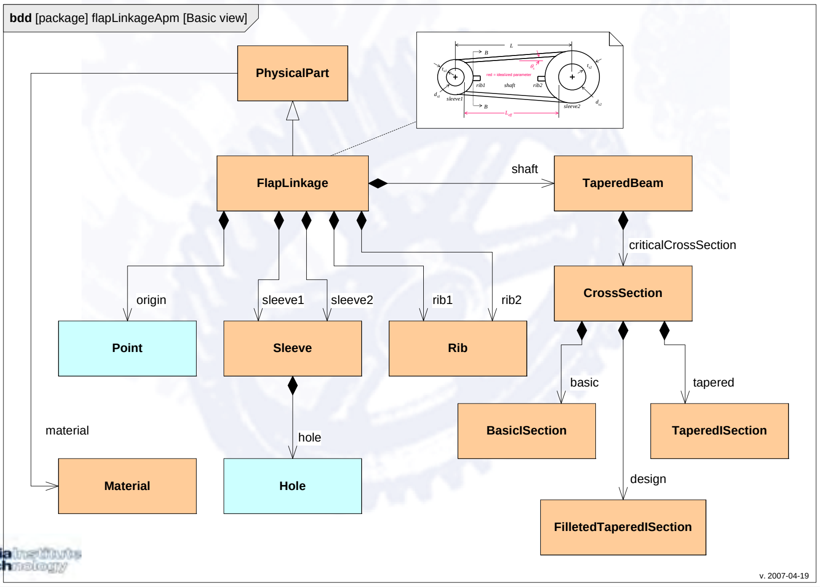
a) Detailed design (CAD model)

d) Sample idealized model: effectiveExtensionalRod

CT-16-14

b0. Flap Linkage Design Model

SysML Block Definition Diagram (bdd) - Basic View

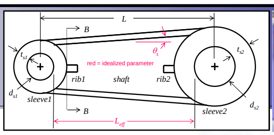


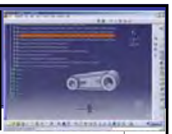
Georgia Institute of Technology

CT-16-15

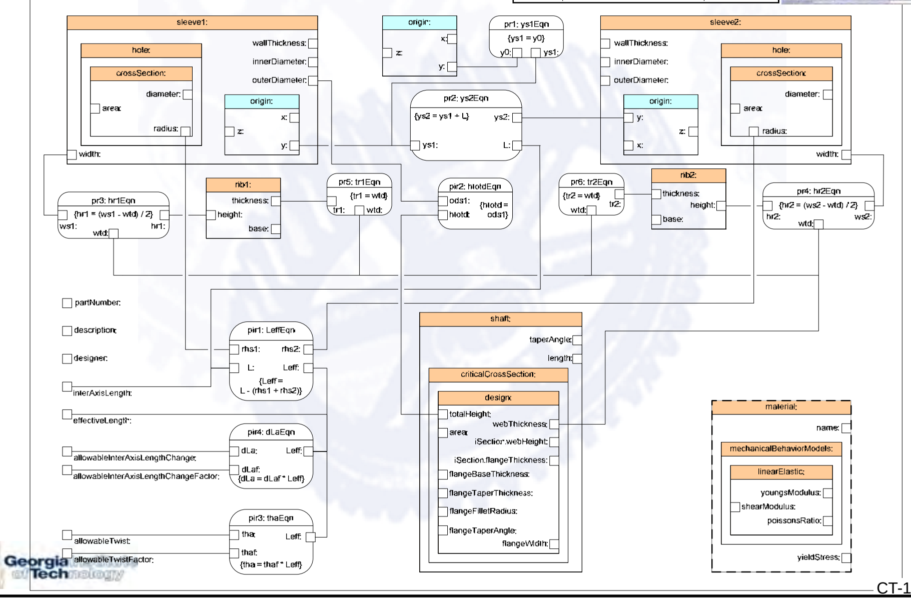
b0 Flap Linkage Design Model

SysML Parametric Diagram (par)





par [block] FlapLinkage [Definition view: primary design and idealization relations]



Georgia Institute of Technology

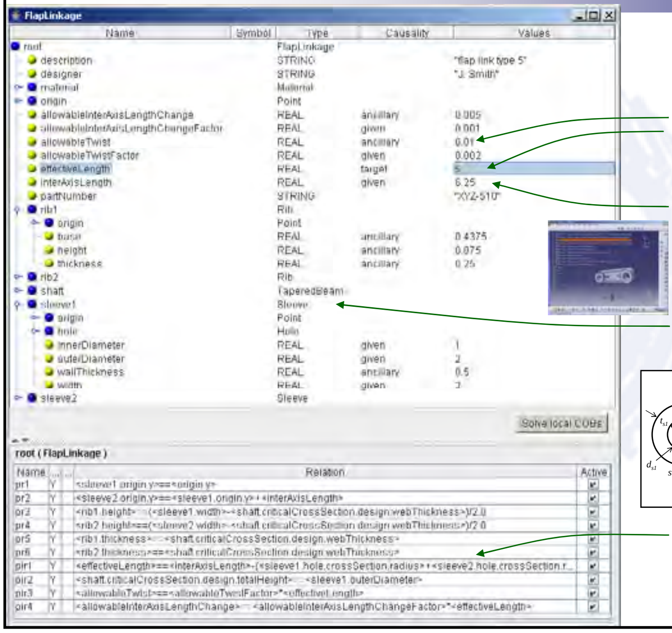
CT-16-16

b0 FlapLinkage Parametric Design Relations

Name	*Constraint*	*Constraint substitution form*
pr1:	*{ys1 = y0}*	*{sleeve1.origin.y = origin.y}*
pr2:	*{ys2 = ys1 + L}*	*{sleeve2.origin.y = sleeve1.origin.y + interAxisLength}*
pr3:	*{hr1 = (ws1 - wtd) / 2}*	*{rib1.height = (sleeve1.width - shaft.criticalCrossSection.design.webThickness)/2}*
pr4:	*{hr2 = (ws2 - wtd) / 2}*	*{rib2.height = (sleeve2.width - shaft.criticalCrossSection.design.webThickness)/2}*
pr5:	*{tr1 = wtd}*	*{rib1.thickness = shaft.criticalCrossSection.design.webThickness }*
pr6:	*{tr2 = wtd}*	*{rib2.thickness = shaft.criticalCrossSection.design.webThickness }*
pir1:	*{Leff = L - (rhs1 + rhs2)}*	*{effectiveLength = interAxisLength - (sleeve1.hole.crossSection.radius + sleeve2.hole.crossSection.radius)}*
pir2:	*{htotd = ods1}*	*{shaft.shaft.criticalCrossSection.design.totalHeight = sleeve1.outerDiameter}*
pir3:	*{tha = thaf * Leff}*	*{allowableTwist = allowableTwistFactor * effectiveLength}*
pir4:	*{dLa = dLaf * Leff}*	*{allowableInterAxisLengthChange = allowableInterAxisLengthChangeFactor * effectiveLength}*

b0. Design Template Instance: Flap Linkage XYZ-150

Executable parametric model in XaiTools *COB browser—an object-oriented spreadsheet.*



FlapLinkage

Name	Symbol	Type	Causality	Values
root		FlapLinkage		
description		STRING		"flap link type 5"
designer		STRING		"J. Smith"
material		Material		
origin		Point		
allowableInterAxisLengthChange		REAL	ancillary	0.005
allowableInterAxisLengthChangeFactor		REAL	given	0.001
allowableTwist		REAL	ancillary	0.01
allowableTwistFactor		REAL	given	0.002
effectiveLength		REAL	target	5
interAxisLength		REAL	given	6.25
partNumber		STRING		"XYZ-510"
rib1		Rib		
origin		Point		
base		REAL	ancillary	0.4375
height		REAL	ancillary	0.075
thickness		REAL	ancillary	0.25
rib2		Rib		
shaft		TaperedBeam		
sleeve1		Sleeve		
origin		Point		
hole		Hole		
innerDiameter		REAL	given	1
outerDiameter		REAL	given	2
wallThickness		REAL	ancillary	0.5
width		REAL	given	2
sleeve2		Sleeve		

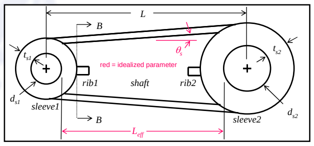
root (FlapLinkage)

Name	...	...	Relation	Active
pr1	Y		<sleeve1.origin.y>==<origin.y>	✓
pr2	Y		<sleeve2.origin.y>==<sleeve1.origin.y>+<interAxisLength>	✓
pr3	Y		<rib1.height>==(<sleeve1.width>-<shaft.criticalCrossSection.design.webThickness>)/2.0	✓
pr4	Y		<rib2.height>==(<sleeve2.width>-<shaft.criticalCrossSection.design.webThickness>)/2.0	✓
pr5	Y		<rib1.thickness>==<shaft.criticalCrossSection.design.webThickness>	✓
pr6	Y		<rib2.thickness>==<shaft.criticalCrossSection.design.webThickness>	✓
pir1	Y		<effectiveLength>==<interAxisLength>-(<sleeve1.hole.crossSection.radius>+<sleeve2.hole.crossSection.r...	✓
pir2	Y		<shaft.criticalCrossSection.design.totalHeight>==<sleeve1.outerDiameter>	✓
pir3	Y		<allowableTwist>==<allowableTwistFactor>*<effectiveLength>	✓
pir4	Y		<allowableInterAxisLengthChange>==<allowableInterAxisLengthChangeFactor>*<effectiveLength>	✓

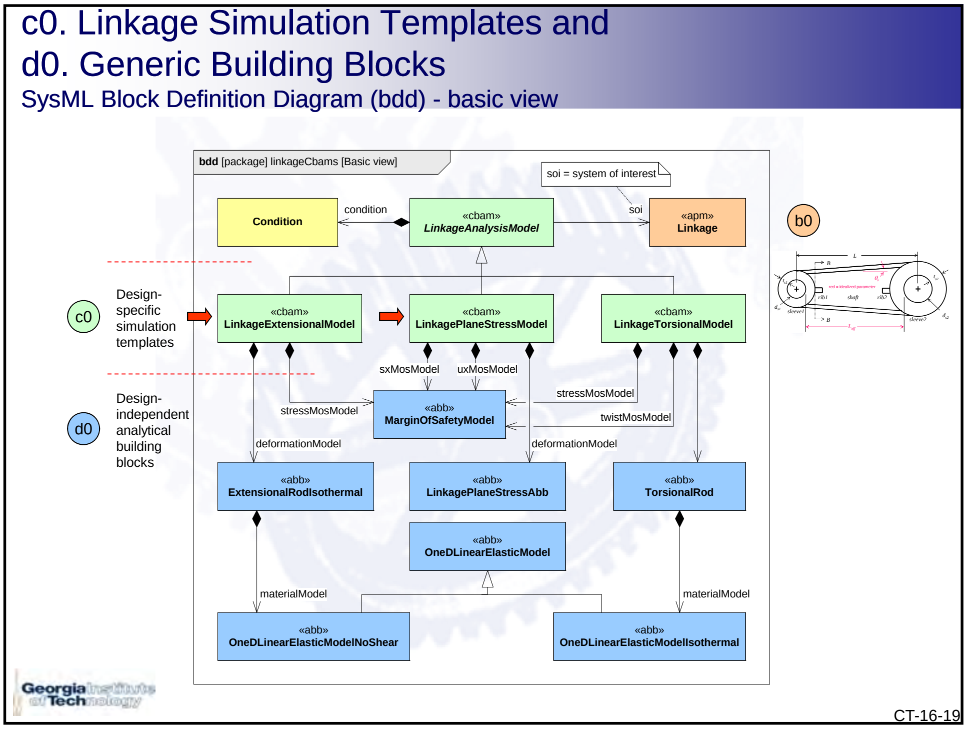
Computed outputs
(targets and ancillary outputs)

Detailed design inputs
from CAD and requirements
(givens)

Design features
(object-oriented structure)



Parametric design relationships
(multi-directional)

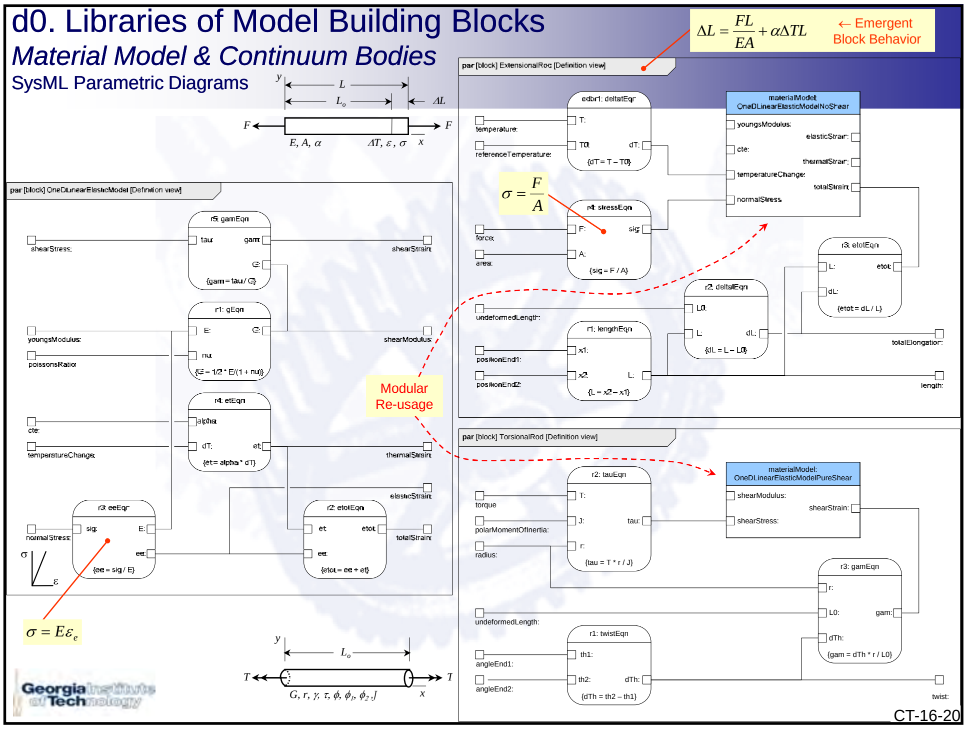


c0. Linkage Simulation Templates and d0. Generic Building Blocks

SysML Block Definition Diagram (bdd) - basic view

bdd [package] linkageCbams [Basic view]
soi = system of interest
Condition
condition
«cbam» LinkageAnalysisModel
soi
«apm» Linkage
b0
c0
Design-specific simulation templates
«cbam» LinkageExtensionalModel
«cbam» LinkagePlaneStressModel
«cbam» LinkageTorsionalModel
sxMosModel
uxMosModel
stressMosModel
«abb» MarginOfSafetyModel
stressMosModel
twistMosModel
d0
Design-independent analytical building blocks
deformationModel
deformationModel
«abb» ExtensionalRodIsothermal
«abb» LinkagePlaneStressAbb
«abb» TorsionalRod
«abb» OneDLinearElasticModel
materialModel
materialModel
«abb» OneDLinearElasticModelNoShear
«abb» OneDLinearElasticModelIsothermal
L
B
rib1
shaft
rib2
sleeve1
sleeve2
red = idealized parameter
Georgia Institute of Technology

CT-16-19



d0. Libraries of Model Building Blocks

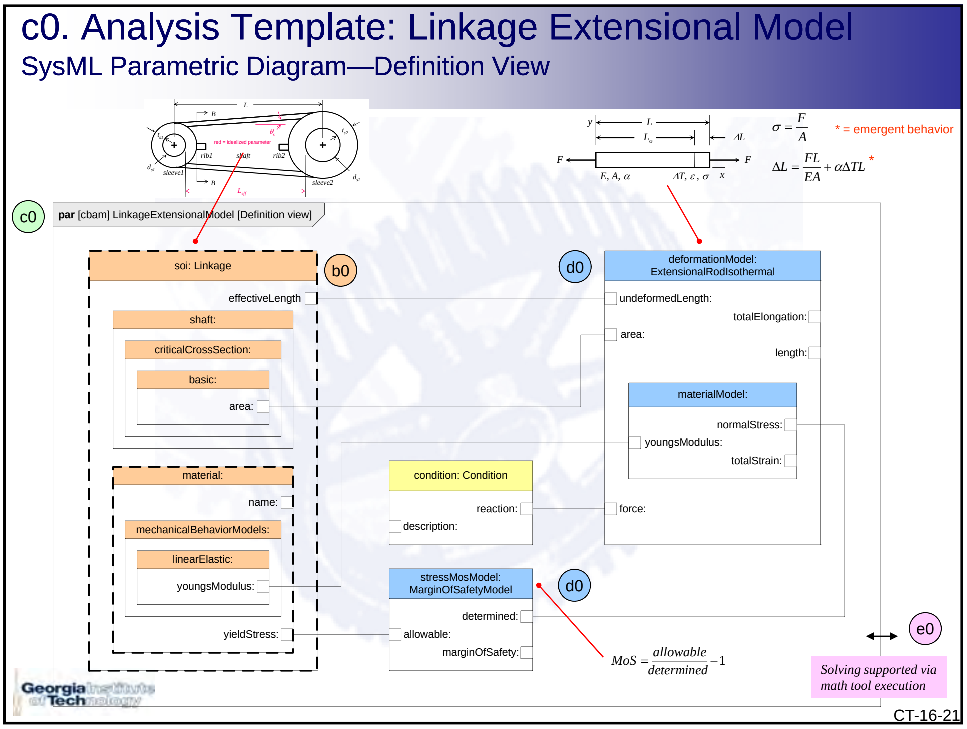
Material Model & Continuum Bodies

SysML Parametric Diagrams

$$\Delta L = \frac{FL}{EA} + \alpha \Delta TL$$

← Emergent Block Behavior

CT-16-20



c0. Analysis Template: Linkage Extensional Model

SysML Parametric Diagram—Definition View

$$\sigma = \frac{F}{A}$$

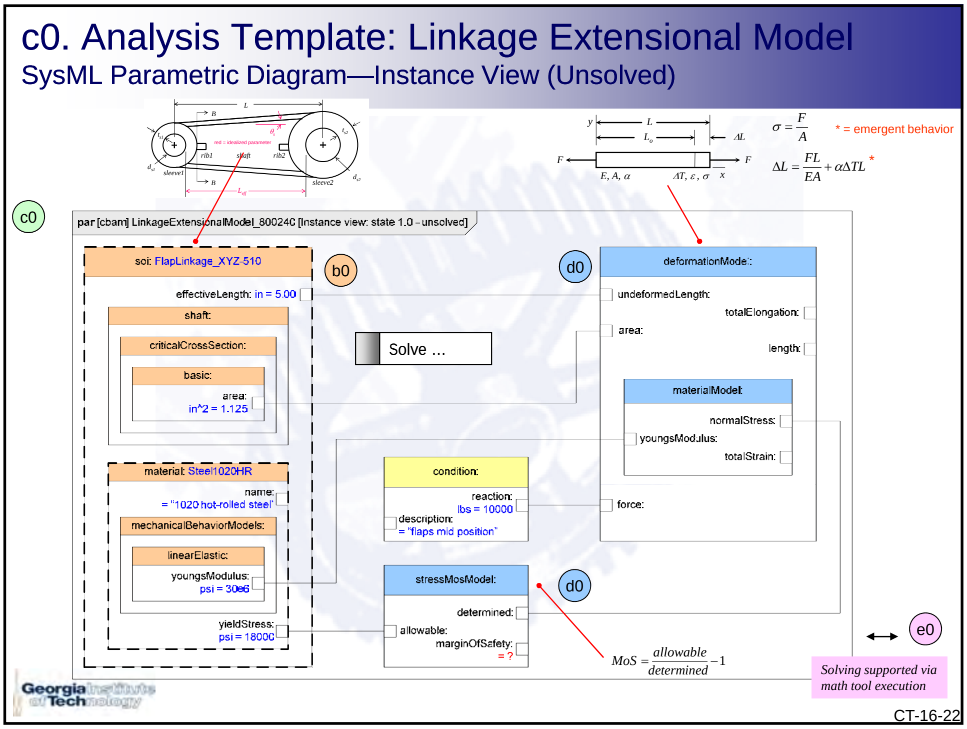
$$\Delta L = \frac{FL}{EA} + \alpha \Delta T L \; *$$

* = emergent behavior

$$MoS = \frac{allowable}{determined} - 1$$

Solving supported via math tool execution

CT-16-21



c0. Analysis Template: Linkage Extensional Model

SysML Parametric Diagram—Instance View (Unsolved)

$$\sigma = \frac{F}{A}$$

$$\Delta L = \frac{FL}{EA} + \alpha \Delta T L \; *$$

* = emergent behavior

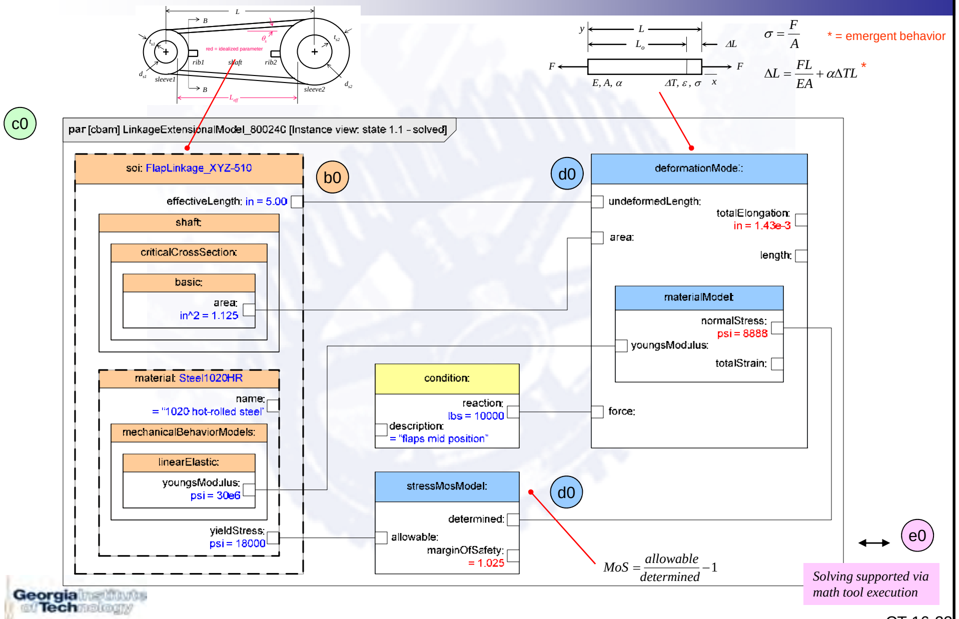
$$MoS = \frac{allowable}{determined} - 1$$

Solving supported via math tool execution

CT-16-22

c0. Analysis Template: Linkage Extensional Model

SysML Parametric Diagram—Instance View (Solved)



$$\sigma = \frac{F}{A}$$

$$\Delta L = \frac{FL}{EA} + \alpha \Delta TL \ ^{*}$$

* = emergent behavior

$$MoS = \frac{allowable}{determined} - 1$$

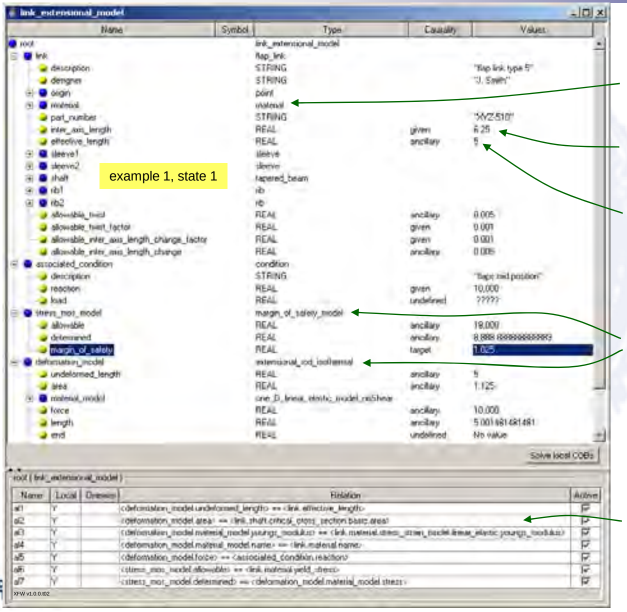
Solving supported via math tool execution

v. 2005-12-19

CT-16-23

c0. Analysis Template Instance: Linkage Extensional Model

Executable parametric model in XaiTools *Browser—an object-oriented spreadsheet.*



```
Name                                      Symbol  Type                               Causality  Values
root                                              link_extensional_model
  link                                            flap_link
    description                                   STRING                                        "flap link type 5"
    designer                                      STRING                                        "J. Smith"
    origin                                        point
    material                                      material
    part_number                                   STRING                                        "XYZ-510"
    inter_axis_length                             REAL                               given      6.25
    effective_length                              REAL                               ancillary  5
    sleeve1                                       sleeve
    sleeve2                                       sleeve
    shaft                                         tapered_beam
    rib1                                          rib
    rib2                                          rib
    allowable_twist                               REAL                               ancillary  0.005
    allowable_twist_factor                        REAL                               given      0.001
    allowable_inter_axis_length_change_factor     REAL                               given      0.001
    allowable_inter_axis_length_change            REAL                               ancillary  0.005
  associated_condition                            condition
    description                                   STRING                                        "flaps mid position"
    reaction                                      REAL                               given      10,000
    load                                          REAL                               undefined  ?????
  stress_mos_model                                margin_of_safety_model
    allowable                                     REAL                               ancillary  18,000
    determined                                    REAL                               ancillary  8,888.888888888889
    margin_of_safety                              REAL                               target     1.025
  deformation_model                               extensional_rod_isothermal
    undeformed_length                             REAL                               ancillary  5
    area                                          REAL                               ancillary  1.125
    material_model                                one_D_linear_elastic_model_noShear
    force                                         REAL                               ancillary  10,000
    length                                        REAL                               ancillary  5.001481481481
    end                                           REAL                               undefined  No value
```

root (link_extensional_model)

Name	Local	Dimension	Relation	Active
a1	Y		<deformation_model.undeformed_length> == <link.effective_length>	☑
a2	Y		<deformation_model.area> == <link.shaft.critical_cross_section.basic.area>	☑
a3	Y		<deformation_model.material_model.youngs_modulus> == <link.material.stress_strain_model.linear_elastic.youngs_modulus>	☑
a4	Y		<deformation_model.material_model.name> == <link.material.name>	☑
a5	Y		<deformation_model.force> == <associated_condition.reaction>	☑
a6	Y		<stress_mos_model.allowable> == <link.material.yield_stress>	☑
a7	Y		<stress_mos_model.determined> == <deformation_model.material_model.stress>	☑

XFW v1.0.0.t02

Library data for materials

Detailed CAD data from CATIA

Idealized analysis features in APM

Modular generic building blocks

Focal Point of CAD-CAE Integration

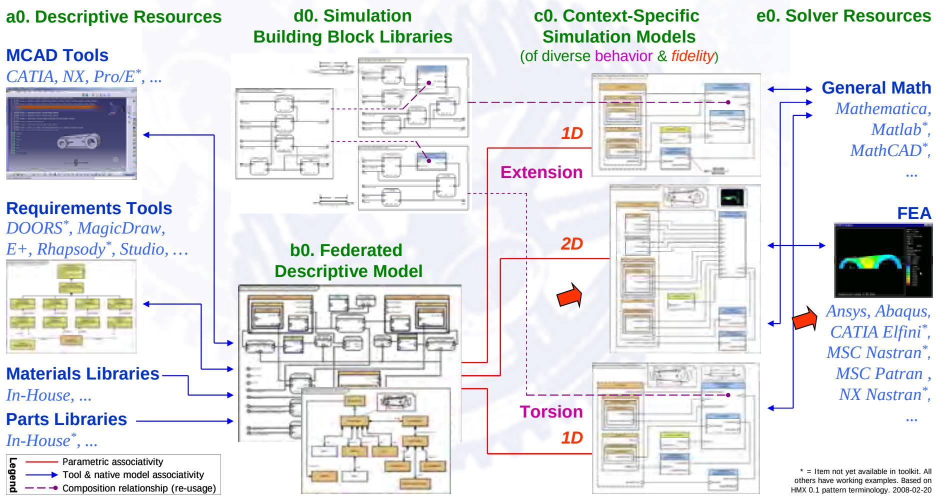
Explicit multi-directional associativity between design & analysis

CT-16-24

MIM Panorama for Mechanical Design

Flap Linkage Model—A Benchmark Design-Analysis Example

Interoperability Panorama View (per MIM 0.1 terminology)



a0. Descriptive Resources

MCAD Tools
CATIA, NX, Pro/E, …*

Requirements Tools
DOORS, MagicDraw, E+, Rhapsody*, Studio, …*

Materials Libraries
In-House, …

Parts Libraries
In-House, …*

d0. Simulation Building Block Libraries

b0. Federated Descriptive Model

c0. Context-Specific Simulation Models
(of diverse behavior & *fidelity*)

e0. Solver Resources

General Math
Mathematica, Matlab, MathCAD*, …*

FEA
Ansys, Abaqus, CATIA Elfini, MSC Nastran*, MSC Patran , NX Nastran*, …*

Legend
- Parametric associativity
- Tool & native model associativity
- Composition relationship (re-usage)

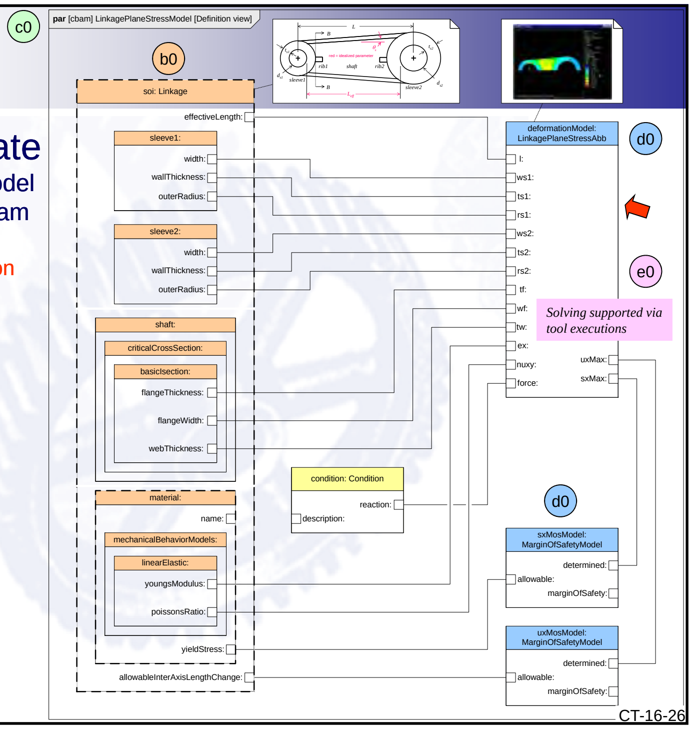
* = Item not yet available in toolkit. All others have working examples. Based on HMX 0.1 pattern terminology. 2008-02-20

CT-16-25

c0. FEA-based Analysis Template

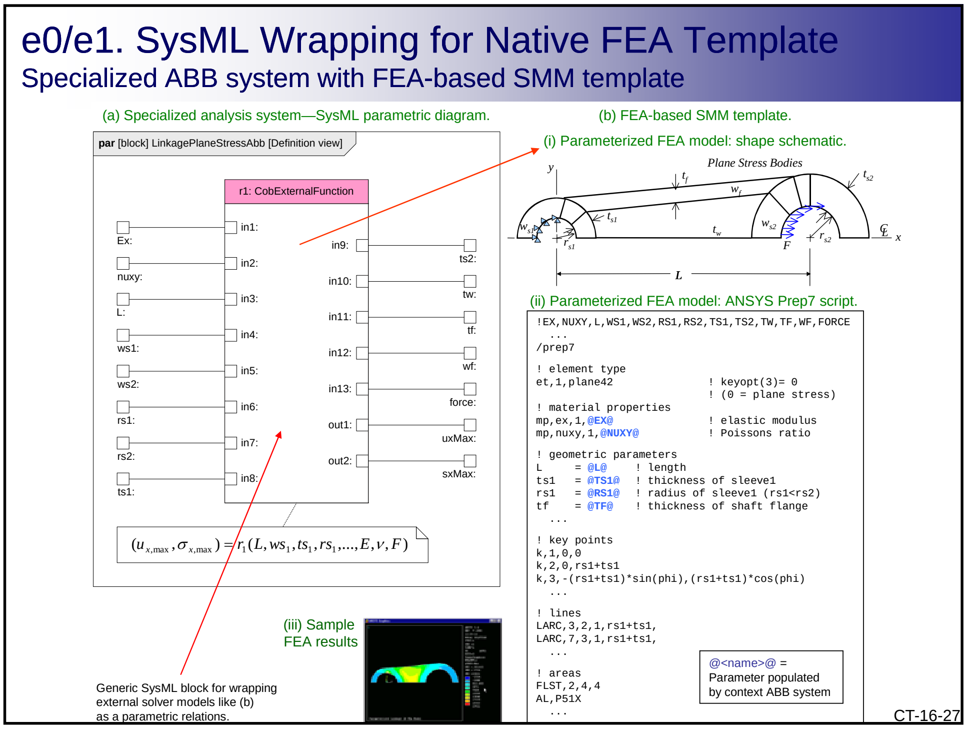
Linkage Plane Stress Model
SysML Parametric Diagram

(see example implementation in CT-13.2)



Solving supported via tool executions

CT-16-26



e0/e1. SysML Wrapping for Native FEA Template

Specialized ABB system with FEA-based SMM template

(a) Specialized analysis system—SysML parametric diagram.

$(u_{x,\max}, \sigma_{x,\max}) = r_1(L, ws_1, ts_1, rs_1, ..., E, \nu, F)$

Generic SysML block for wrapping external solver models like (b) as a parametric relations.

(iii) Sample FEA results

(b) FEA-based SMM template.

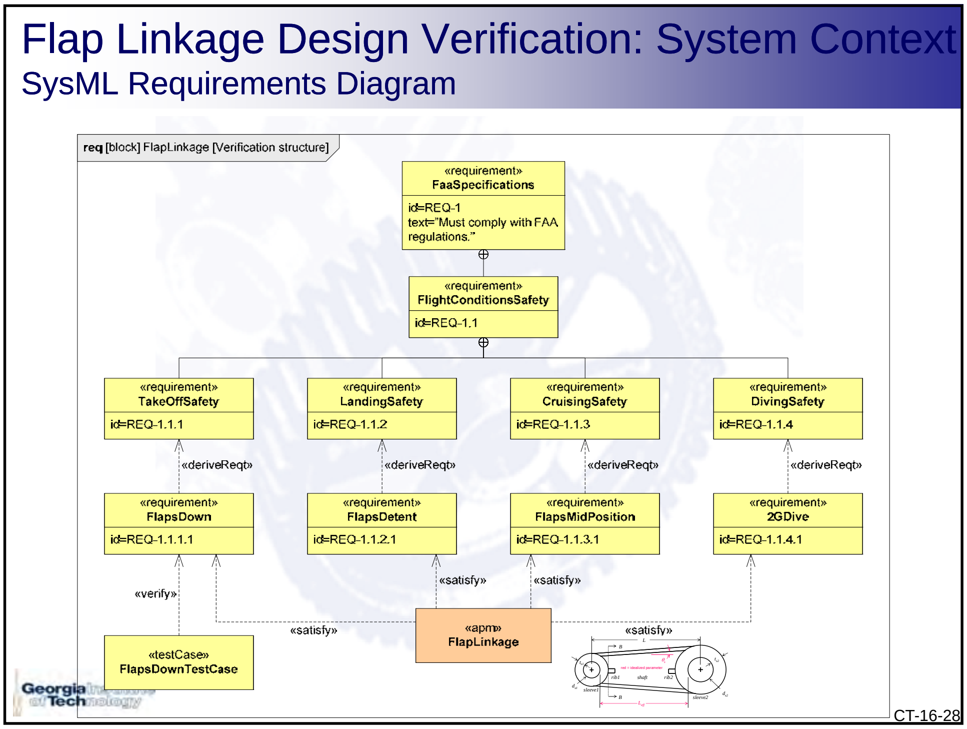
(i) Parameterized FEA model: shape schematic.

(ii) Parameterized FEA model: ANSYS Prep7 script.

```
!EX,NUXY,L,WS1,WS2,RS1,RS2,TS1,TS2,TW,TF,WF,FORCE
  ...
/prep7

! element type
et,1,plane42                  ! keyopt(3)= 0
                              ! (0 = plane stress)
! material properties
mp,ex,1,@EX@                  ! elastic modulus
mp,nuxy,1,@NUXY@              ! Poissons ratio

! geometric parameters
L     = @L@    ! length
ts1   = @TS1@  ! thickness of sleeve1
rs1   = @RS1@  ! radius of sleeve1 (rs1<rs2)
tf    = @TF@   ! thickness of shaft flange
  ...

! key points
k,1,0,0
k,2,0,rs1+ts1
k,3,-(rs1+ts1)*sin(phi),(rs1+ts1)*cos(phi)
  ...

! lines
LARC,3,2,1,rs1+ts1,
LARC,7,3,1,rs1+ts1,
  ...

! areas
FLST,2,4,4
AL,P51X
  ...
```

@<name>@ = Parameter populated by context ABB system

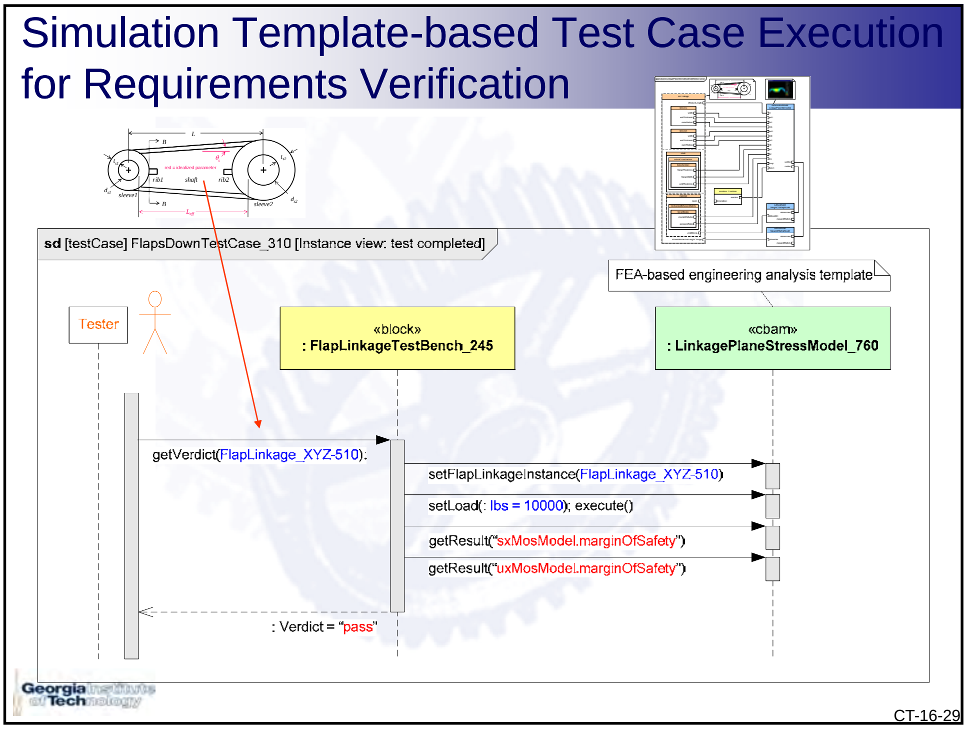
CT-16-27



Flap Linkage Design Verification: System Context

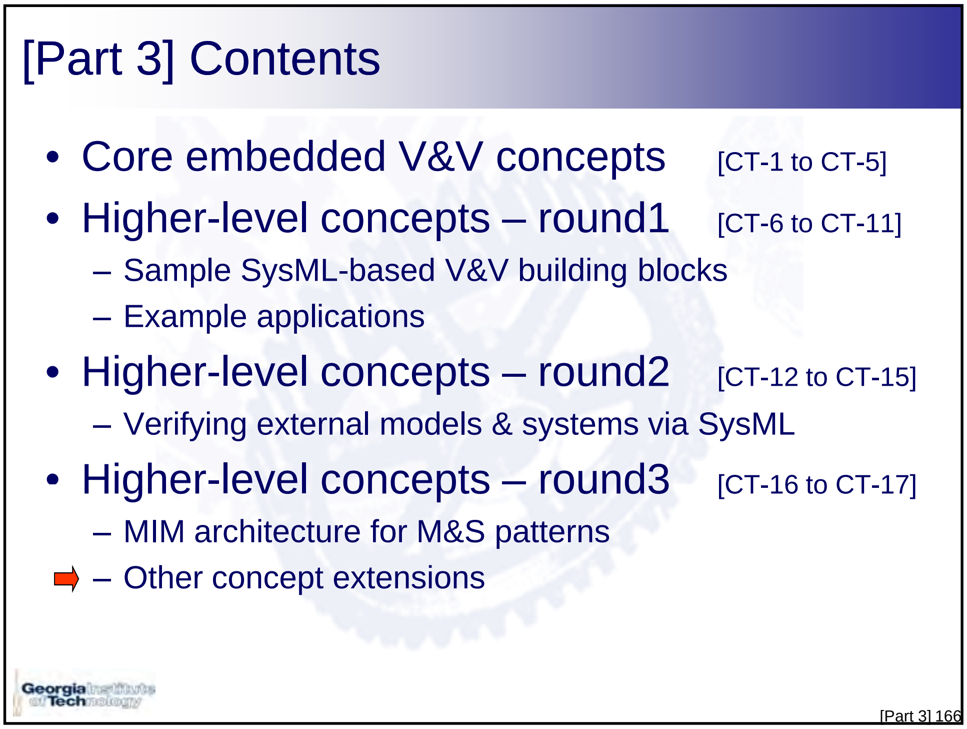
SysML Requirements Diagram

CT-16-28



Simulation Template-based Test Case Execution for Requirements Verification

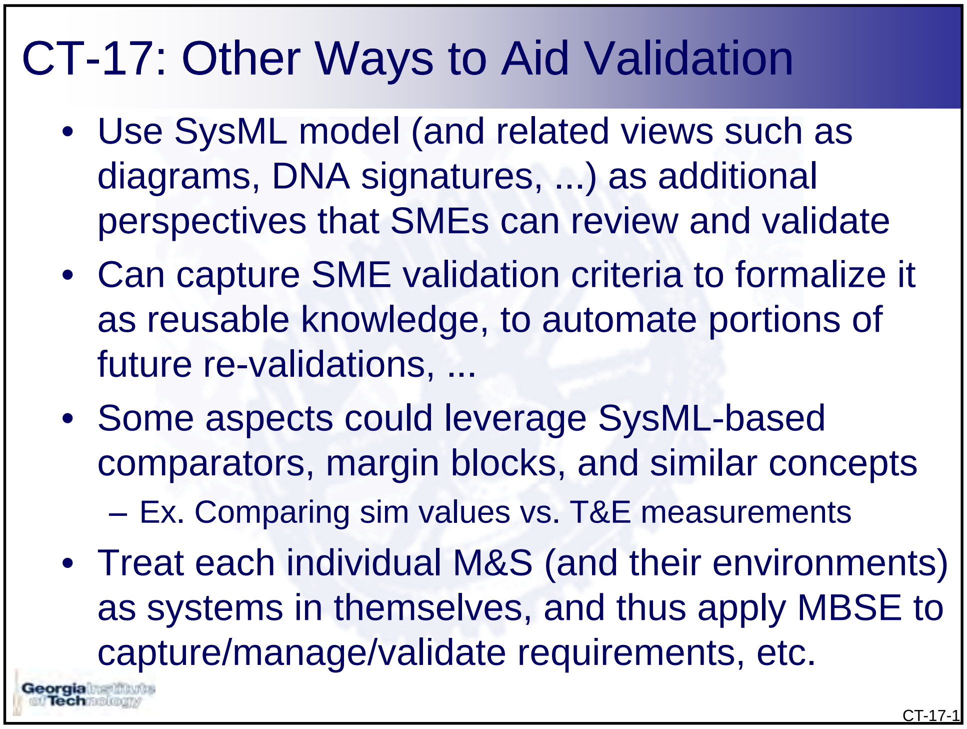
sd [testCase] FlapsDownTestCase_310 [Instance view: test completed]
FEA-based engineering analysis template
Tester
«block»
: FlapLinkageTestBench_245
«cbam»
: LinkagePlaneStressModel_760
getVerdict(FlapLinkage_XYZ-510):
setFlapLinkageInstance(FlapLinkage_XYZ-510)
setLoad(: lbs = 10000); execute()
getResult("sxMosModel.marginOfSafety")
getResult("uxMosModel.marginOfSafety")
: Verdict = "pass"
Georgia Institute of Technology
CT-16-29



[Part 3] Contents

- Core embedded V&V concepts [CT-1 to CT-5]
- Higher-level concepts – round1 [CT-6 to CT-11]
  - Sample SysML-based V&V building blocks
  - Example applications
- Higher-level concepts – round2 [CT-12 to CT-15]
  - Verifying external models & systems via SysML
- Higher-level concepts – round3 [CT-16 to CT-17]
  - MIM architecture for M&S patterns
  - ➡ Other concept extensions

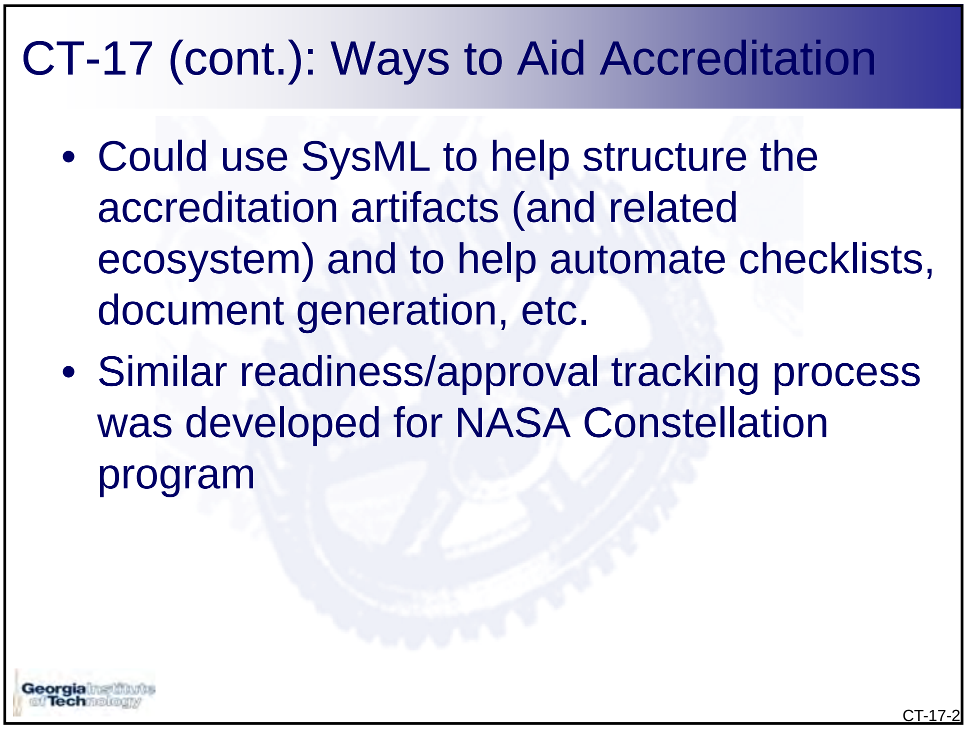
Georgia Institute of Technology
[Part 3] 166



CT-17: Other Ways to Aid Validation

- Use SysML model (and related views such as diagrams, DNA signatures, ...) as additional perspectives that SMEs can review and validate
- Can capture SME validation criteria to formalize it as reusable knowledge, to automate portions of future re-validations, ...
- Some aspects could leverage SysML-based comparators, margin blocks, and similar concepts
  - Ex. Comparing sim values vs. T&E measurements
- Treat each individual M&S (and their environments) as systems in themselves, and thus apply MBSE to capture/manage/validate requirements, etc.

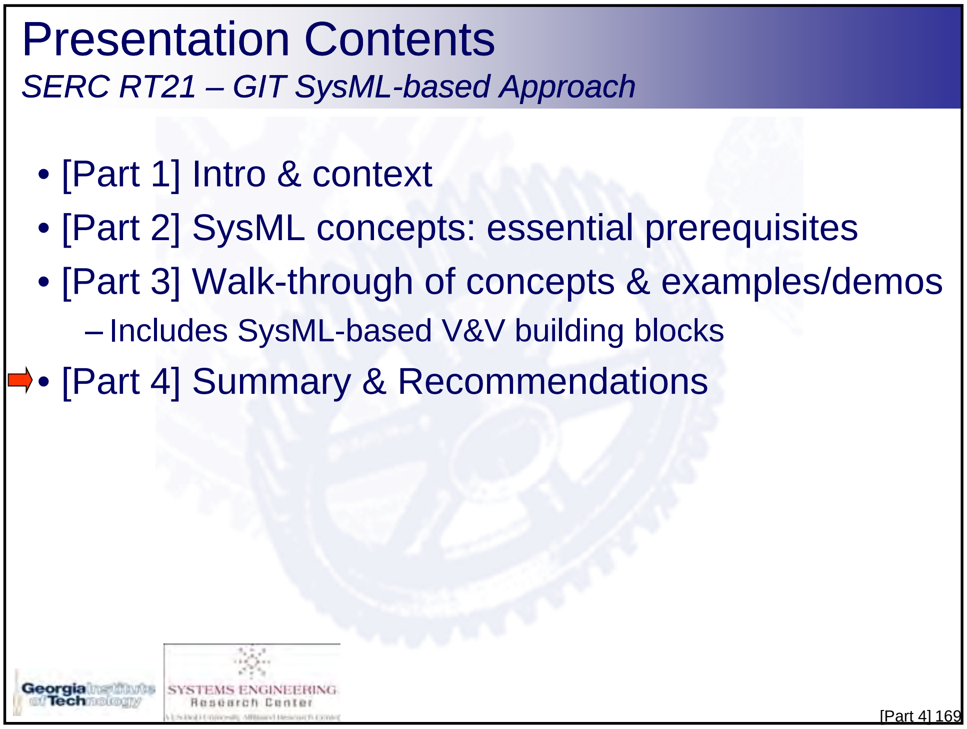
CT-17-1



CT-17 (cont.): Ways to Aid Accreditation

- Could use SysML to help structure the accreditation artifacts (and related ecosystem) and to help automate checklists, document generation, etc.
- Similar readiness/approval tracking process was developed for NASA Constellation program

CT-17-2

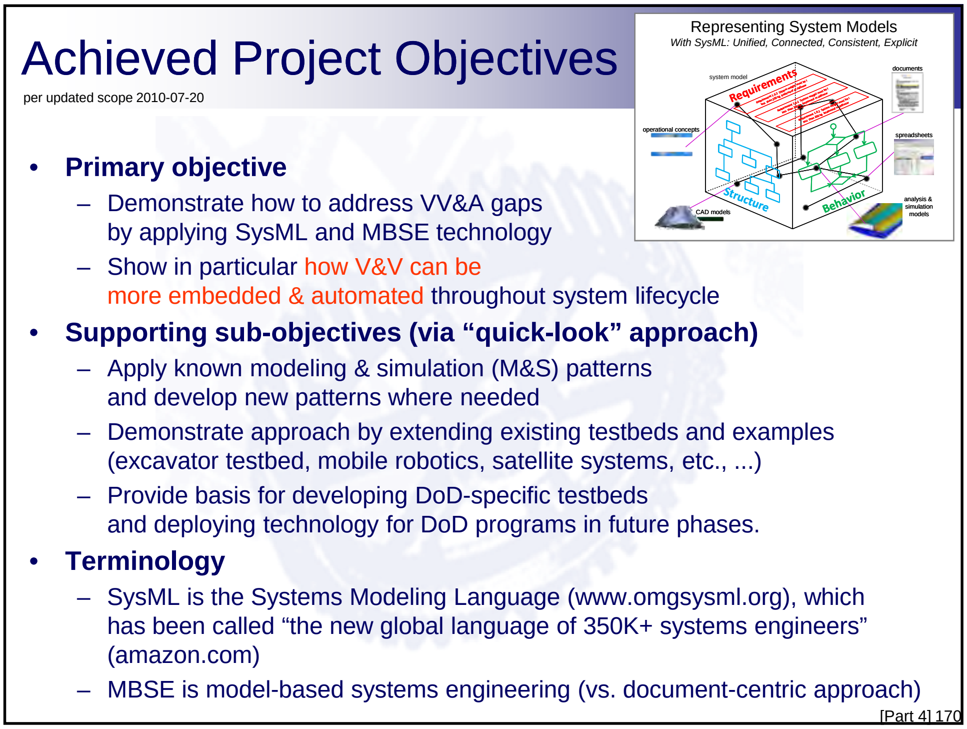


Presentation Contents

SERC RT21 – GIT SysML-based Approach

- [Part 1] Intro & context
- [Part 2] SysML concepts: essential prerequisites
- [Part 3] Walk-through of concepts & examples/demos
  - Includes SysML-based V&V building blocks
- ➡ [Part 4] Summary & Recommendations

[Part 4] 169



Achieved Project Objectives

per updated scope 2010-07-20

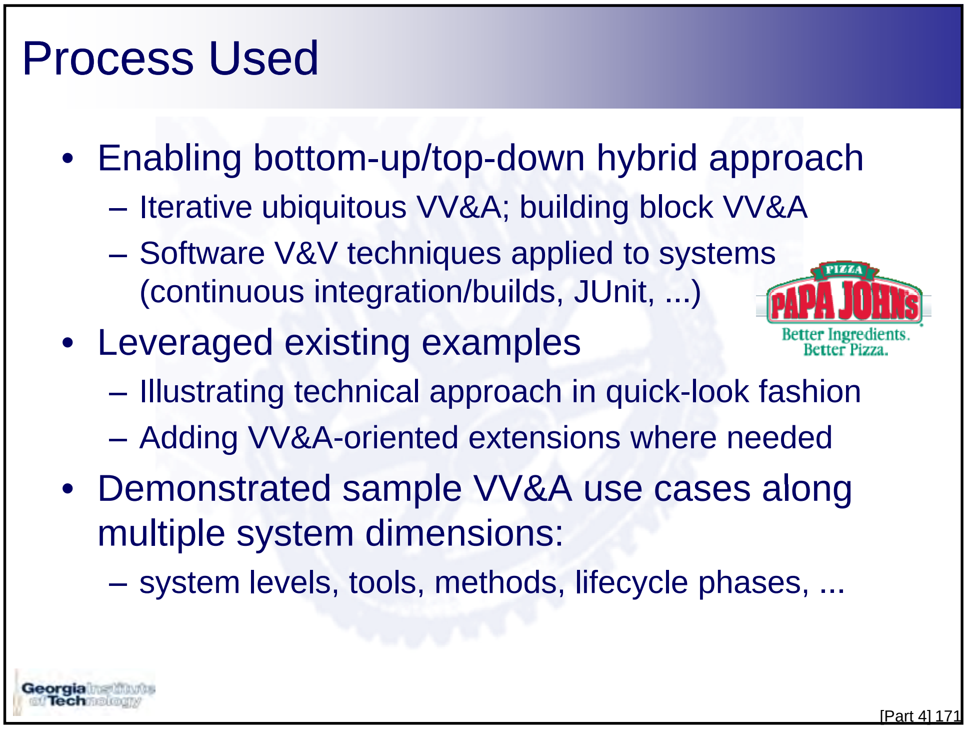
Representing System Models

With SysML: Unified, Connected, Consistent, Explicit

system model
Requirements
Structure
Behavior
operational concepts
CAD models
documents
spreadsheets
analysis & simulation models

- **Primary objective**
  - Demonstrate how to address VV&A gaps by applying SysML and MBSE technology
  - Show in particular how V&V can be more embedded & automated throughout system lifecycle
- **Supporting sub-objectives (via “quick-look” approach)**
  - Apply known modeling & simulation (M&S) patterns and develop new patterns where needed
  - Demonstrate approach by extending existing testbeds and examples (excavator testbed, mobile robotics, satellite systems, etc., ...)
  - Provide basis for developing DoD-specific testbeds and deploying technology for DoD programs in future phases.
- **Terminology**
  - SysML is the Systems Modeling Language (www.omgsysml.org), which has been called “the new global language of 350K+ systems engineers” (amazon.com)
  - MBSE is model-based systems engineering (vs. document-centric approach)

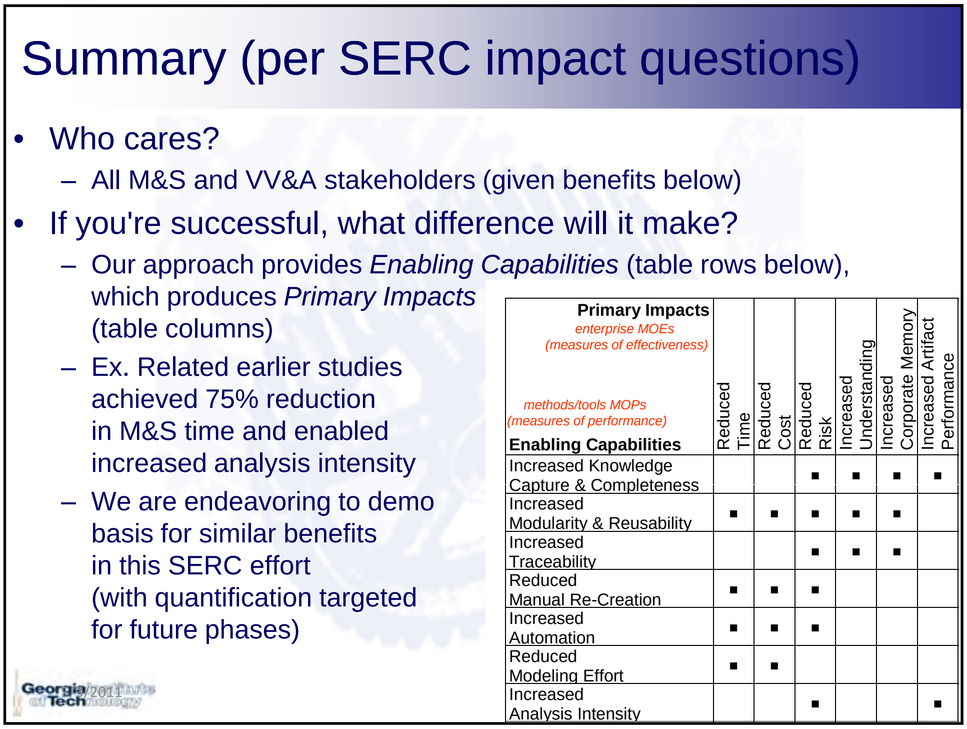
[Part 4] 170



Process Used

- Enabling bottom-up/top-down hybrid approach
  - Iterative ubiquitous VV&A; building block VV&A
  - Software V&V techniques applied to systems (continuous integration/builds, JUnit, ...)
- Leveraged existing examples
  - Illustrating technical approach in quick-look fashion
  - Adding VV&A-oriented extensions where needed
- Demonstrated sample VV&A use cases along multiple system dimensions:
  - system levels, tools, methods, lifecycle phases, ...

[Part 4] 171

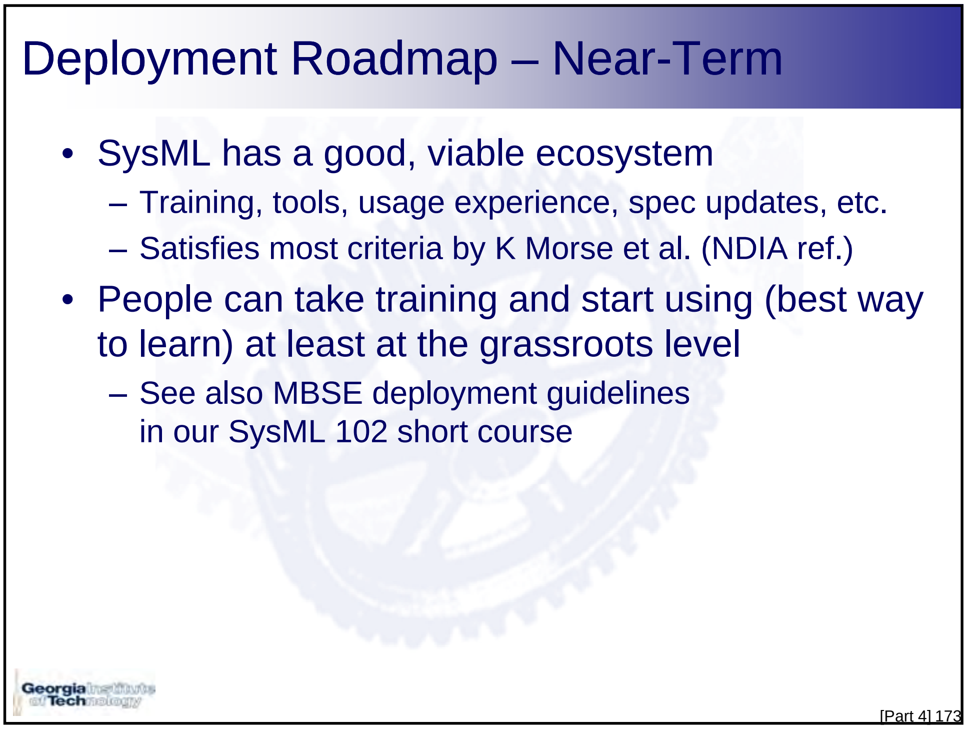


Summary (per SERC impact questions)

- Who cares?
  - All M&S and VV&A stakeholders (given benefits below)
- If you're successful, what difference will it make?
  - Our approach provides *Enabling Capabilities* (table rows below), which produces *Primary Impacts* (table columns)
  - Ex. Related earlier studies achieved 75% reduction in M&S time and enabled increased analysis intensity
  - We are endeavoring to demo basis for similar benefits in this SERC effort (with quantification targeted for future phases)

Primary Impacts *enterprise MOEs (measures of effectiveness)* *methods/tools MOPs (measures of performance)* **Enabling Capabilities**	Reduced Time	Reduced Cost	Reduced Risk	Increased Understanding	Increased Corporate Memory	Increased Artifact Performance
Increased Knowledge Capture & Completeness			■	■	■	■
Increased Modularity & Reusability	■	■	■	■	■	
Increased Traceability			■	■	■	
Reduced Manual Re-Creation	■	■	■			
Increased Automation	■	■	■			
Reduced Modeling Effort	■	■				
Increased Analysis Intensity			■			■

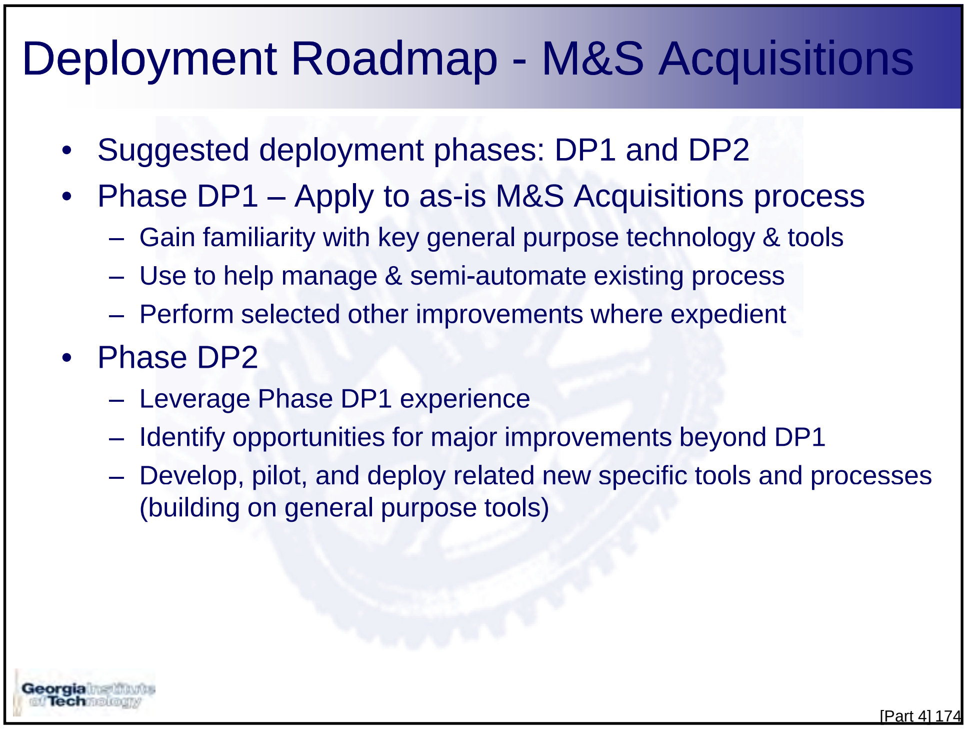
2011



Deployment Roadmap – Near-Term

- SysML has a good, viable ecosystem
  - Training, tools, usage experience, spec updates, etc.
  - Satisfies most criteria by K Morse et al. (NDIA ref.)
- People can take training and start using (best way to learn) at least at the grassroots level
  - See also MBSE deployment guidelines
    in our SysML 102 short course

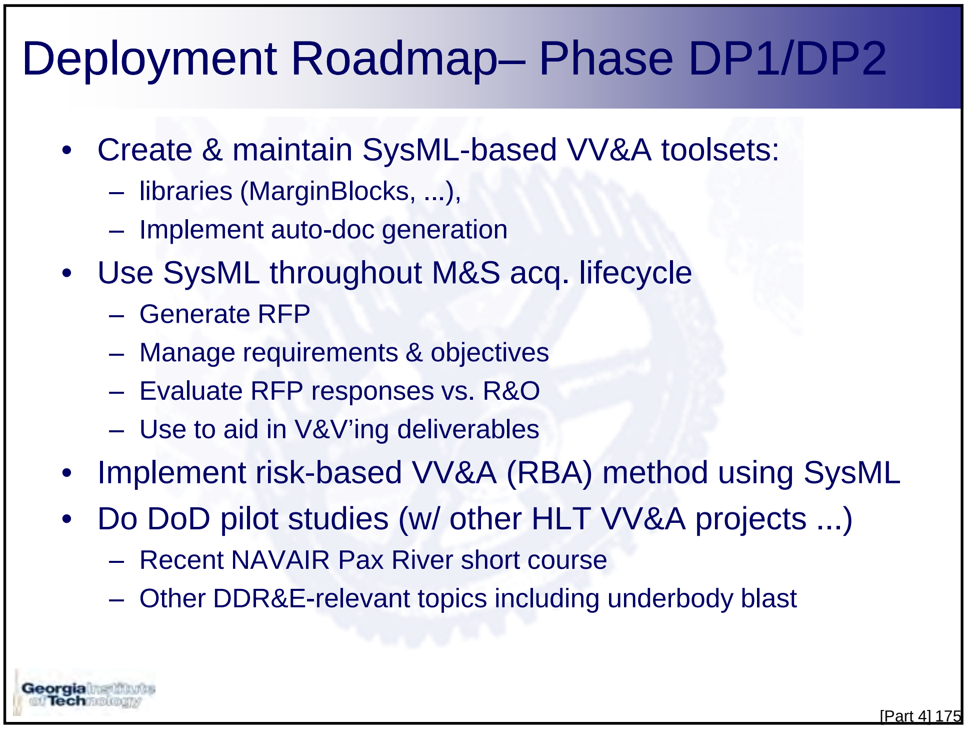
[Part 4] 173



Deployment Roadmap - M&S Acquisitions

- Suggested deployment phases: DP1 and DP2
- Phase DP1 – Apply to as-is M&S Acquisitions process
  - Gain familiarity with key general purpose technology & tools
  - Use to help manage & semi-automate existing process
  - Perform selected other improvements where expedient
- Phase DP2
  - Leverage Phase DP1 experience
  - Identify opportunities for major improvements beyond DP1
  - Develop, pilot, and deploy related new specific tools and processes (building on general purpose tools)

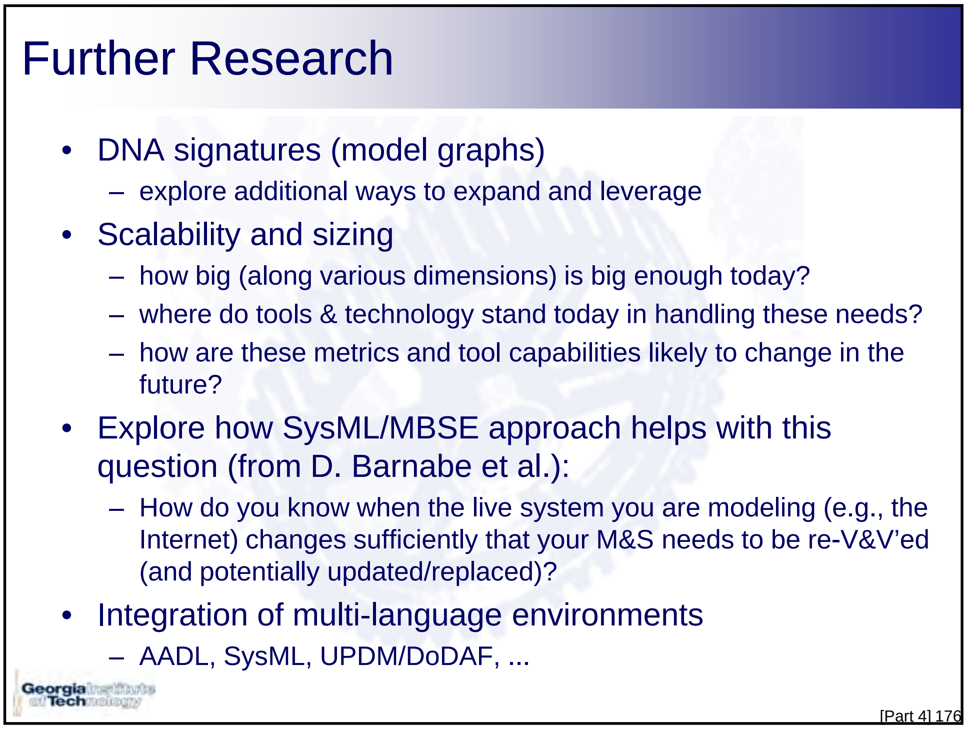
[Part 4] 174



Deployment Roadmap– Phase DP1/DP2

- Create & maintain SysML-based VV&A toolsets:
  - libraries (MarginBlocks, ...),
  - Implement auto-doc generation
- Use SysML throughout M&S acq. lifecycle
  - Generate RFP
  - Manage requirements & objectives
  - Evaluate RFP responses vs. R&O
  - Use to aid in V&V'ing deliverables
- Implement risk-based VV&A (RBA) method using SysML
- Do DoD pilot studies (w/ other HLT VV&A projects ...)
  - Recent NAVAIR Pax River short course
  - Other DDR&E-relevant topics including underbody blast

[Part 4] 175



Further Research

- DNA signatures (model graphs)
  - explore additional ways to expand and leverage
- Scalability and sizing
  - how big (along various dimensions) is big enough today?
  - where do tools & technology stand today in handling these needs?
  - how are these metrics and tool capabilities likely to change in the future?
- Explore how SysML/MBSE approach helps with this question (from D. Barnabe et al.):
  - How do you know when the live system you are modeling (e.g., the Internet) changes sufficiently that your M&S needs to be re-V&V'ed (and potentially updated/replaced)?
- Integration of multi-language environments
  - AADL, SysML, UPDM/DoDAF, ...

[Part 4] 176